# Universidad de las Ciencias Informáticas

## Facultad 15


**Título:** Componentes Visuales que extienden la arquitectura de dominio específico del GINA

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

**Autores:** Lázaro Manuel Gil Martínez
Jorge Alexis Del Castillo Palma

**Tutor:** Msc. Julio Cesar Díaz Vera

**Ciudad de La Habana, Mayo de 2010**

**"Año 52 de la Revolución"**

## DECLARACIÓN DE AUTORÍA

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

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

Lázaro Manuel Gil Martínez

_____

Firma del Autor

Jorge Alexis Del Castillo Palma

_____

Firma del Autor

Julio Cesar Díaz Vera

_____

Firma del Tutor

*"... las cualidades morales suben de precio cuando están realzadas por las cualidades inteligentes..."*

*José Martí*

# Agradecimientos

*En primer lugar quisiera agradecer a mis madres Ruth y Sahilys por hacer de mi la persona que soy hoy, por brindarme su apoyo, paciencia y las fuerzas necesarias para lograr un título universitario.*

*A mi papá por brindarme su ejemplo y depositar tanta confianza en mí.*

*A mi tía Ileana que se ha comportado como una madre para mí y también me ha apoyado tanto.*

*A mi familia de San Antonio, Matanzas y La Habana, mi tía Ñica, Iván, Mayi, Alberto, Mirtica, Popolo, Yeney, Marcos, y mis primos Lando, Rubén, Lázaro, los jimaguas, la Tatica, Mary, Yani, y Claudia, que me han liberado en incontables momentos de la estresante vida universitaria.*

*A mi compañero de tesis por su paciencia y dedicación para la realización de este trabajo.*

*A mi tutor Julio por encaminarme no sólo en la realización de este trabajo, sino en todo mi trabajo en el proyecto desde un principio.*

*A Ricardo por enseñarme muchas de las cosas que sé y ayudarme en la realización de este trabajo.*

*A mi peque, es decir mi novia, por quererme y apoyarme tanto, ser tan paciente conmigo y trabajar a mi lado para lograr la confección de este documento.*

*A mis amigos Carlos, Oscar, Dairelis, Lulú, y muchos otros, por las tantas comidas y fiestas compartidas.*

*A Fran y Lumi por brindarme tanta ayuda y lo más importante su amistad.*

*A todos muchas gracias.*

*Lázaro Manuel Gil Martínez*

# Agradecimientos

*Esta tesis es el broche que cierra una etapa de mi vida como estudiante, etapa que si bien ha requerido de esfuerzo y dedicación de mi parte, no hubiese sido posible su finalización sin la ayuda desinteresada de todas y cada una de las personas que a continuación citaré.*

*Muchas son las personas especiales a las que me gustaría agradecer. Algunas se encuentran presentes y otras desafortunadamente no, unas viven y vivirán siempre en mis recuerdos y mi corazón. Sin importar donde estén y si alguna vez llegan a leer esto, quiero darles las gracias por formar parte de mi vida, por brindarme amor, apoyo, amistad, ánimo, compañía, por sus consejos y bendiciones.*

*Primero y antes que nada, dar gracias a Dios y a "Mi madre espiritual", la Virgen de la Caridad del Cobre, por estar conmigo en cada paso que doy, por fortalecer mi corazón e iluminar mi mente y por poner en mi camino a aquellas personas que han sido mi soporte y compañía durante toda mi vida.*

*A mi familia en general, por proveerme la estabilidad emocional y económica para afrontar la tarea del estudio sin otras preocupaciones.*

*A mi mamá, a quien debo todo lo que he logrado en la vida, mi formación como profesional y hombre de bien, gracias por creer en mí, por tu amor, por tus consejos, por tu guía; si he llegado a ser lo que hoy soy es todo gracias a tí, te quiero.*

*A mi papá, gracias por ser mi modelo a seguir, por tu amor y tu apoyo incondicional en cualquier sentido.*

*A mi hermana, por ser más que mi hermana mi mejor amiga.*

*A mi abuela Mamelia por su apoyo y sus bendiciones, a mi tío Lázaro, A Viki, mis hermanos Iván y Lazarito por ser la mejor familia que alguien podría tener.*

*A mis familias de La Habana (María, Yane y Alejandro) y de Manzanillo, por su apoyo incondicional.*

*Al Mayor Julián, por ayudarme a llegar donde estoy.*

*A mi compañero de Tesis, por su gran dedicación a este trabajo y su amistad.*

*A mi tutor, por su gran paciencia y su ayuda en la realización de este trabajo y a lo largo de estos años de estudiante universitario, muchas gracias Julio.*

*A mis compañeros en general, "Los salvajes" (Ricardo, Carlos, Oscar, Yoidel), "Los tigres" (Humberto, Granda, El Dennis, Prieto), "Las conejas" (La Jabá, Taylén, Leydis) y especialmente a Annaliet por su apoyo, su ayuda y todo lo que vivimos durante estos años de Universidad.*

*A otros que no quisiera dejar de mencionar, Dairelis, Christian, Leevan, Sucel, mis compañeros de grupo, de proyecto, y todos con los que he convivido en estos años de Universidad.*

*A todos, muchas gracias.*

*Alex*

*Le dedico este trabajo a toda mi familia, en especial a mi hermano, que le sirva como ejemplo, y a mi abuelo Manolo que estoy seguro estuviese orgulloso de verme dar otro paso en la vida.*

*Lázaro Manuel Gil Martínez*

*Este trabajo está dedicado especialmente con mucho amor y cariño a la memoria de mi abuelo Papancho, donde quiera que se encuentre sé que estará muy orgulloso de ver mi realización profesional y como hombre de bien. Que sepa que si algo me animó siempre a seguir hacia adelante sin importar obstáculos a lo largo de esta carrera es el sentimiento de que él siempre se encuentra conmigo en cada paso que doy en la vida, motivación suficiente para superarme cada día más y luchar para que la vida me depare un futuro mejor, que estoy seguro es lo que él querría que sucediese.*

*Alex*

## RESUMEN

La Universidad de las Ciencias Informáticas (UCI) en conjunto con la Aduana General de la República (AGR) se encuentran enmarcados en el desarrollo de una aplicación para la Gestión Integral Aduanera (GINA). El avance de este proyecto ha sido golpeado en gran medida por las demoras en cuanto a la creación de las interfaces de la capa de presentación, debido a la poca capacitación de los desarrolladores para realizar las tareas asignadas y la repetición continua del código implementado.

El siguiente trabajo propone implementar un conjunto de componentes visuales reutilizables para la capa de interfaz de usuario con el objetivo de agilizar el proceso de desarrollo de las interfaces de la aplicación y disminuir la curva de aprendizaje del framework ExtJS utilizado para diseñar las interfaces. Con el fin de realizar la tarea propuesta se identificaron los requerimientos de interfaz que presentaban un comportamiento común, para encapsular dichos comportamientos en la implementación de los componentes.

## ÍNDICE

## ÍNDICE DE TABLAS



## ÍNDICE DE FIGURAS

## CAPÍTULO 1: INTRODUCCIÓN

En nuestros días el vertiginoso avance de la informática y las comunicaciones, con su mayor exponente: la Internet, y la cada vez más creciente demanda de aplicaciones de calidad que dan solución a las más específicas necesidades, ha hecho que las técnicas tradicionales de diseño e implementación de las aplicaciones comiencen a ser insuficientes, ya que los sistemas son cada vez más complejos, deben ser construidos en menos tiempo y deben cumplir con los estándares más altos de calidad, por lo que un nuevo enfoque de desarrollo se hace necesario.

La presente investigación se enmarca dentro de los desarrollos realizados para la Aduana General de la República y continúa un grupo de trabajos cuyo objetivo final es dotar a la arquitectura base de estas aplicaciones con capacidad para hacer uso del enfoque de desarrollo de software orientado a componentes y la utilización de buenas prácticas de desarrollo propuestas en esquemas de dominio específico. En última instancia se pretenden generalizar los procesos de implementación, estandarizando las soluciones de desarrollo.

El blanco primario de desarrollo para esta etapa del trabajo está centrado en la capa de interfaz de usuario, tomando en cuenta el alto impacto que tendría sobre las soluciones desarrolladas aumentar el rendimiento de los desarrolladores de dicha capa, que en estos momentos registran los peores índices dentro del proyecto (ver Tabla 1), y una disminución, por tanto, de los tiempos de desarrollo de los productos a implementar en el Departamento de Soluciones para la Aduana (DSA), el cual es el objetivo principal de este trabajo.

Trabajos previos realizados en el Departamento de Soluciones para la Aduana han propuesto la creación de componentes visuales que extiendan la arquitectura de dominio específica del sistema aduanal como un mecanismo que facilite la implementación de las aplicaciones. A pesar de que se ha propuesto un mecanismo para identificar los componentes necesarios a desarrollar, este no ha sido llevado a la práctica y por tanto no han sido detectados y mucho menos implementados dichos componentes.

Esto provoca demora en los tiempos de implementación de los productos a desarrollar, lo que influye negativamente en el cumplimiento de los cronogramas establecidos en el Departamento de Soluciones para la Aduana, con respecto a implementación y entrega de los productos elaborados en dicho departamento. De ahí la necesidad de que se generalicen los procesos de implementación de la interfaz visual en los proyectos para así lograr reducir el tiempo de ejecución de los mismos.

Esta investigación pretende accionar especialmente sobre el trabajo de los desarrolladores de la capa de presentación del Departamento de Soluciones para la Aduana, específicamente para el aprovechamiento por parte de estos de las muchas ventajas que brinda el enfoque de desarrollo de software orientado a componentes, ventajas tales como la reutilización de código y la estandarización de las soluciones, contribuyendo así a un mayor desempeño de dichos desarrolladores, aumentando la velocidad de implementación, la calidad y el rendimiento de las aplicaciones a desarrollar, así como la facilidad de mantenimiento y actualización.

La idea principal por la que se rige esta investigación expresa lo siguiente: *implementando un conjunto de componentes visuales que incluyan lógica de negocio y que extiendan las funcionalidades del framework de dominio específico sobre el que se implementa el GINA, dichos componentes debidamente documentados e implementados sobre la base de un estándar de codificación, entonces se reducirá de manera notable la repetición de código y se facilitará el desarrollo de aplicaciones contribuyendo directamente al aumento del desempeño de los implementadores de la capa de presentación y por consecuencia a la disminución de los tiempos de desarrollo de la aplicaciones*.

## Formulación del problema

En los anteriores acápites se expone la situación existente en la Universidad de las Ciencias Informáticas, específicamente en el Departamento de Soluciones para la Aduana con respecto al tiempo que demoran en desarrollarse los productos pertenecientes al Sistema de Gestión Integral de Aduanas (GINA), el cuál es la herramienta informática que se utiliza en la Aduana General de la República para informatizar los

diversos procesos aduanales, como se puede apreciar, dicha situación presenta un grupo de elementos negativos, evidenciados durante el desarrollo de los distintos proyectos asociados al Departamento de Soluciones para la Aduana, lo cual crea las bases para el desarrollo de trabajos de investigación cuyos objetivos estén orientados a minimizar la repercusión de dichos elementos negativos en los compromisos pactados por la Universidad.

De manera particular el presente trabajo pretende abordar esta temática centrándose en el siguiente problema ¿Cómo disminuir los tiempos de desarrollo y estandarizar las soluciones para la arquitectura de dominio específico del GINA?

Por lo antes expuesto, el **objetivo general** de este trabajo es desarrollar un conjunto de componentes visuales que extiendan la arquitectura de dominio específico del GINA.

**El objeto de estudio** se centra en la arquitectura de dominio específico sobre el que se implementa el GINA y en componentes visuales que extiendan dicha arquitectura.

El **campo de acción** está enmarcado en el proceso de implementación de componentes visuales para la arquitectura del GINA.

Para dar cumplimiento al objetivo general de este trabajo se determinó que era necesario dar cumplimiento a los siguientes **objetivos específicos**:

- Identificar los componentes visuales.
- Implementar los componentes visuales.
- Demostrar la viabilidad de la implementación.

| Caso de uso | Fecha planificada | Fecha concluyó |
|---|---|---|
| **CUModificarDatos** | 20/enero/2008 | 24/enero/2008 |
| **CUPendientesaAceptacion** | 22/enero/2008 | 25/enero/2008 |
| **CUResultadosVerificacion** | 26/enero/2008 | 28/enero/2008 |
| **CUCrearFicha** | 29/enero/2008 | 5/febrero/2008 |
| **CUCrearUnidad** | 15/febrero/2008 | 19/febrero/2008 |
| **CUConfeccionarModificarEstructura** | 17/febrero/2008 | 20/febrero/2008 |
| **CUAsignarPatrimonio** | 19/febrero/2008 | 21/febrero/2008 |
| **CUInsertarTrabajador** | 23/marzo/2008 | 25/marzo/2008 |
| **CURenovarContrato** | 24/marzo/2008 | 30/marzo/2008 |
| **CUCrearContratodeAdiestramiento** | 3/abril/2008 | 10/abril/2008 |
| **CUReportarPendientesAnexo** | 6/abril/2008 | 7/abril/2008 |

Tabla 1 Cumplimiento de los cronogramas de desarrollo (Alvarez Garcia, 2009)

El presente documento consta de 4 capítulos:

**Capítulo 1:**

Es la introducción al trabajo, donde se aborda el tema de la investigación, por qué surge la misma, la situación en la que surge, y a quien va destinada.

**Capítulo 2:**

Fundamentación teórica: En este capítulo se realiza un estudio del estado del arte y un análisis de las tecnologías y herramientas utilizadas en esta investigación, de modo que permita lograr un entendimiento de los aspectos más importantes del domino del problema.

**Capítulo 3:**

Se realiza el análisis y la descripción de la solución que dará respuesta al problema planteado al principio del trabajo.

**Capitulo 4**:

Se presenta el análisis de un caso de estudio teniendo en cuenta diferentes parámetros que demuestran la factibilidad del uso de los componentes en el desarrollo de la capa de presentación.

# CAPÍTULO 2: MARCO TEÓRICO

## 2.1 Desarrollo de Software basado en Componentes

La creciente necesidad de realizar aplicaciones complejas en cortos períodos de tiempo, disminuyendo a la vez el esfuerzo necesario tanto en fuerza de trabajo como el coste total del proyecto, está favoreciendo el avance de lo que se conoce como Desarrollo de Software Basado en Componentes (DSBC). Esta disciplina se apoya en componentes software ya desarrollados, que son combinados adecuadamente para satisfacer los requisitos del sistema, entendiendo por componente "una unidad de composición de aplicaciones software que posee un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes, de forma independiente en tiempo y espacio" (F. Bertoa, Troya, & Vallecillo).

La idea que sustenta esta forma de desarrollar descansa sobre un aforismo que ronda alrededor de la informática desde hace unos años…"todo el software que se necesita en el mundo ya ha sido codificado antes".

Construir una aplicación se convierte por tanto en la búsqueda, adaptación y ensamblaje de piezas prefabricadas, desarrolladas en su mayoría por terceros. Bajo este nuevo planteamiento, cobran especial interés los procesos de búsqueda y selección de los componentes apropiados para construir las aplicaciones. En este sentido, la tecnología de componentes ofrece ya soluciones robustas para muchos de los problemas que se plantean en la construcción de grandes sistemas de software y, de hecho, vivimos inmersos en una creciente "componentización" del software. (F. Bertoa, Troya, & Vallecillo).

Dicha disciplina cuenta actualmente con un creciente interés, tanto desde el punto de vista académico como desde el industrial, en donde la demanda de mecanismos y herramientas de desarrollo basados en componentes es cada día mayor.

En general, el desarrollo de software basado en componentes puede verse como una extensión natural de la programación orientada a objetos dentro del ámbito de los sistemas abiertos[1] y distribuidos[2].

Este paradigma se basa en el uso de los componentes software como entidades básicas del modelo.

Los principios fundamentales cuando se diseña un componente son que este debe ser:

- Reusable: Los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas.
- Sin contexto específico: Los componentes son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los datos deben ser pasadas al componente en vez de incluirlos o permitir al componente acceder a ellos.
- Extensible: El componente puede ser extendido desde un componente existente para crear un nuevo comportamiento.
- Encapsulado: Los componentes exponen interfaces que permiten al programa usar su funcionalidad sin revelar detalles internos, detalles del proceso o estado.

[1] Un sistema abierto es un sistema distribuido en el que se intentan conseguir, por lo menos parcialmente los objetivos de los sistemas distribuidos y hacer que las interfaces entre sus componentes respeten un conjunto amplio y completo de normas sobre comunicaciones, programación, presentación e interfaces entre aplicaciones y servicios del sistema aceptadas internacionalmente. (Falgueras Campderrich, 2003)

[2] Hablamos de entorno distribuido cuando el software de una aplicación se ejecuta repartido entre varios ordenadores de una red; entonces también decimos que el software en cuestión es distribuido. (Falgueras Campderrich, 2003)

- Independiente: Los componentes están diseñados para tener una dependencia mínima de otros componentes, por lo que pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas.

Numerosas son las características que aporta la programación orientada a componentes frente a la programación orientada a objetos tradicional. Entre ellas, el desarrollo de los componentes de forma independiente del contexto en donde serán ejecutados, la reutilización por composición (frente a herencia, al tratarse de entidades "binarias" o "cajas negras"), la introspección (facilidad para interrogar al componente sobre sus propiedades y métodos de forma dinámica, normalmente mediante el uso de reflexión), nuevas formas de comunicación (como los "eventos" y las comunicaciones asíncronas frente a los rudimentarios mecanismos de los objetos), el enlazado dinámico y composición tardía, etc.

El paradigma de ensamblar componentes y escribir código para hacer que estos funcionen posee varias ventajas:

- Reutilización del software: Una de las principales ventajas del desarrollo de software basado en componentes se basa en la "reutilización" de los mismos. De esta forma, los componentes se diseñan y desarrollan con el objetivo de poder ser reutilizados en otras aplicaciones, reduciendo el tiempo de desarrollo, mejorando la fiabilidad del producto final (al usar componentes ya probados previamente), y siendo más competitivos en costes.
- Simplifica las pruebas: Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- Simplifica el mantenimiento del sistema: Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- Mayor calidad: Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

De la misma manera, el optar por adquirir componentes hechos por terceros en lugar de desarrollarlos uno mismo provee ventajas tales como:

- Ciclos de desarrollo más cortos: La adición de una pieza dada de funcionalidad tomará días en lugar de meses o años.
- Mejor ROI: Usando correctamente esta estrategia, el retorno sobre la inversión puede ser más favorable que desarrollando los componentes uno mismo.
- Funcionalidad mejorada: Para usar un componente que contenga una pieza de funcionalidad, solamente es necesario entender su naturaleza, más no sus detalles internos. Así, una funcionalidad que sería impráctica de implementar en la empresa, se vuelve ahora completamente asequible.

Este proceso de desarrollo de software ha dado ya sus inicios con implementaciones como la plataforma .net, la cual impulsa la idea de industrializar el software utilizando tecnologías de componentes. Los avances y mejoras presentados en esta plataforma van mucho más allá de las implementaciones iniciales como COM y CORBA, convirtiendo a los componentes .net en verdaderas piezas de ensamblaje.

## 2.2 Arquitecturas de Dominio Específico (DSSA)

El disponer de componentes software no es suficiente para desarrollar aplicaciones, ya provengan estos de un mercado global o sean desarrollados a medida para la aplicación. Un aspecto crítico a la hora de construir sistemas complejos es el diseño de la estructura del sistema y por ello el estudio de la arquitectura de software se ha convertido en una disciplina de especial relevancia en la ingeniería del software.

Se entiende por Arquitectura Software a la representación de alto nivel de la estructura de un sistema o aplicación, dicha arquitectura puede usarse para cualquier dominio de aplicación. Sin embargo, cada

dominio tiene sus peculiaridades específicas y de ahí surgen las llamadas Arquitecturas de Dominio Específico, las cuales incorporan de forma muy particular cada una de las partes que la integran, las interacciones entre ellas, los patrones que supervisan su composición, y las restricciones a la hora de aplicar esos patrones.

Varias son las definiciones que existen en cuanto a arquitectura de dominio específico:

- La arquitectura de dominio específico es un modelo de referencia para una amplia gama de aplicaciones que comparten similares estructura, funcionalidad y comportamiento. No es una aplicación como tal, sino que es, en realidad, un metamodelo que describe cómo se crean los sistemas de instancia específica. ("aplicaciones reales para los clientes reales") (Duffy, 2004).

- Una arquitectura de dominio específico es una estructura genérica, de organización o diseño para sistemas de software en un dominio específico. La arquitectura de dominio específico contiene los diseños que están destinados a satisfacer los requisitos especificados en el modelo de dominio. Una arquitectura de dominio específico puede ser adaptada para crear diseños de sistemas de software dentro de un dominio y también proporciona un marco para la configuración de activos dentro de sistemas de software individuales. (encyclopedia2.thefreedictionary, 2010).

Pero según el marco de esta investigación, la definición más apegada es la siguiente:

Una arquitectura de dominio específico es un conjunto de componentes de software, especializado para un determinado tipo de tarea (de dominio), generalizado para el uso eficaz a través de ese dominio, integrado en una estructura estandarizada (topología) eficaz para la creación de aplicaciones exitosas. (Tracz, 1995).

Las arquitecturas de dominio específico son modelos de arquitectura los cuales son específicos para algún dominio de aplicación. Existen 2 tipos de modelos de dominio específico:

1. Modelos Genéricos

   - Son abstracciones de varios sistemas reales que encapsulan las características principales de estos sistemas.
   - Se pueden utilizar directamente en el diseño.
   - Se derivan de forma ascendente a partir de los sistemas existentes.
   - Pocos modelos genéricos están disponibles al público; las organizaciones que desarrollan estos modelos los ven como una propiedad intelectual necesaria para el desarrollo de futuros sistemas.

Los modelos genéricos usualmente surgen de manera bottom-up, surgen de la práctica y es el tipo de modelo de dominio específico al que se hace referencia en esta investigación.

2. Modelos de Referencia. Estos son modelos idealistas, más abstractos. Proporcionan un significado de información con respecto a sistemas de clases y comparación de diversas arquitecturas.

   - Son modelos abstractos idealizados del dominio que describen a una clase mayor de sistemas.
   - No necesariamente reflejan la arquitectura real de sistemas existentes en el dominio.
   - Se utilizan para comunicar conceptos del dominio y comparar las posibles arquitecturas.
   - Se derivan de forma descendente.

Los modelos de referencia son modelos top-down, surgen de estudios teóricos muy profundos.

Los aspectos clave del enfoque incluyen la reutilización de software basado en la parametrización de los componentes genéricos y la interconexión de los componentes dentro de un marco de solución.

"Con el fin de la reutilización del software, es necesario que exista software reusable." Uno de los dilemas que ha impedido a los desarrolladores la reutilización del software es la falta de artefactos de software a utilizar o la existencia de artefactos que son difíciles de integrar.

Las Arquitecturas de Software de Dominio Específico se han propuesto con el fin de abordar estas cuestiones. Un DSSA no sólo proporciona un marco de trabajo para adaptar componentes de software reutilizables, sino que capta la lógica de diseño y establece un grado de adaptabilidad.

El objetivo del desarrollo de software utilizando las arquitecturas de software de dominio específico es la reducción en el tiempo y costo de producción de sistemas para aplicaciones soportadas dentro de un dominio particular, además de gran aumento a la calidad del producto, facilidad de uso mejorada, y el posicionamiento para la adquisición de futuros negocios.

Los aspectos clave del enfoque incluyen la reutilización de software basado en la parametrización de los componentes genéricos y la interconexión de los componentes dentro de un marco de solución.

La viabilidad de este enfoque depende de la identificación y la comprensión profunda de un dominio seleccionado de aplicaciones. El enfoque DSSA, para ser efectivamente aplicado, requiere de una variedad de herramientas de soporte, con mecanismos de repositorio incluido, facilidades de prototipado, herramientas de análisis y componentes ya desarrollados.

En el Departamento de Soluciones para la Aduana se pretende trabajar bajo esta filosofía, es decir, aplicando el enfoque DSSA, pero por el momento no se han desarrollados los componentes necesarios para establecer un marco de trabajo que funcione bajo este tipo de desarrollo de software.

## 2.3 Arquitectura de Dominio Específico de la Aduana

Para el marco arquitectónico de los sistemas que se desarrollan en el Departamento de Soluciones para la Aduana se ha definido un framework de dominio específico construido a partir de la vinculación de los frameworks de desarrollo Symfony, de PHP, para el caso del controlador y ExtJS, de Javascript; para la capa de presentación, así como Propel para la persistencia de los datos.

A continuación se hace una breve descripción de las características de la arquitectura utilizada en las aplicaciones aduanales.

- Envío de datos:
  - La manera en que se envían los datos entre las capas en el sentido de la vista hacia el controlador puede ser de varias formas como son:
    - El submit de un formulario.
    - Codificando los datos a formato JSON.
    - Por la combinación de ambos.
  - En el sentido inverso, es decir, enviar datos desde el controlador hacia la capa de presentación se hace siempre mediante la codificación de los datos a JSON.
- Estructura de los subsistemas:
  - Cada subsistema contiene la implementación de las vistas que responden a sus propios requerimientos.
  - Existen puntos comunes entre los sistemas y/o subsistemas, en el cual comparten funcionalidades, como es el caso de los componentes visuales desarrollados en este trabajo, para los cuales se crea un subsistema común o toolkit del cual hace uso la aplicación completa.
- Comunicación entre los subsistemas:
  - En el caso de que algún subsistema necesite de información que se maneja en otro subsistema, este último se convierte en un plugin del framework Symfony, lo cual crea un punto común que permite estas funcionalidades de comunicación entre subsistemas.

Aunque puede darse el caso de que la información deba ser consultada a través del script, en tal caso se utiliza un patrón IOC (Inversión of Control), además existe la posibilidad de que la información sea mostrada a través de una interfaz, por lo que el subsistema que va a brindar la información tiene la obligación de construir un controlador para este fin.

- Todos los componentes visuales desarrollados en este trabajo responden al dominio aduanal, por ejemplo, varios de estos componentes manejan tablas de control, sin embargo ninguno funcionaria en un dominio donde no se evaluaran las tablas de control tal y como se hace dentro de las aplicaciones aduanales.

## 2.4 La capa de presentación

De forma particular en el desarrollo de aplicaciones web se ha hecho popular, y además es muy recomendado, la utilización de framework. Una estructura de software compuesta de componentes, personalizables e intercambiables, para el desarrollo de una aplicación y puede ser considerado como una aplicación genérica incompleta y configurable, a la que se le pueden añadir las últimas piezas para construir una aplicación concreta. Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

Los objetivos principales que persigue un framework son:

- Desarrollo rápido.
- Desarrollo estructurado.
- Reutilización de código.
- Disminución del esfuerzo en el desarrollo.
- Aprovechar las funcionalidades ya implementadas.

Al ser Javascript la tecnología imperante en el desarrollo de la capa de presentación en aplicaciones web es innegable la necesidad de contar con framework Javascript que puede definirse como un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de aplicaciones basadas en AJAX, una tecnología para construir páginas web dinámicas del lado del cliente.

Esto es alcanzado más fácilmente con el uso de un framework dedicado a procesar peticiones AJAX. En el artículo donde nació el término "AJAX", J. J. Garret describe ésta tecnología como "un intermediario... entre el usuario y el servidor".

La meta del framework es proveer este motor AJAX y funciones asociadas al servidor y del lado del cliente.

Una librería asiste el trabajo del desarrollador en dos niveles: en el lado del cliente, ofreciendo funciones Javascript para enviar peticiones al servidor y en el lado del servidor, el cual procesa las peticiones, busca información, y la transmite al navegador.

Algunos framework son muy robustos y proveen una biblioteca completa para construir aplicaciones web.

Los frameworks Javascript se ejecutan en el navegador (browser-side) y son ampliamente usados en el desarrollo de aplicaciones AJAX.

### 2.4.1 Frameworks JS

#### 2.4.1.1 ExtJS

Antes de comenzar a examinar ExtJS y cualquier otro framework Javascript primero es necesario 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 plugins 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, este sería el render de la aplicación que controla el cliente y que se encarga de enviar y obtener información del servicio.

ExtJS es un framework para Javascript muy utilizado en el desarrollo de aplicaciones Web con AJAX. Tiene una librería inmensa que permite configurar la interfaces Web de manera semejante a las aplicaciones desktop.

ExtJS empezó siendo un conjunto de librerías y extensiones para YUI (Yahoo! User Inteface). Con el tiempo se convirtió en un framework independiente y a principios de 2007 se creó una compañía para comercializar y dar soporte del framework Ext. De esta forma ExtJS tiene dos tipos de licencias, LGPL y comercial. (http://extjs.com/, 2010).

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.).

ExtJS posee un modelo de componentes jerárquico que facilita la reutilización de cada uno de los componentes que forman esta librería y promueven la extensión de los mismos con vista a desarrollos específicos de alto grado de personalización.

Usar un motor de render como ExtJS conlleva varios beneficios, tales como:

- 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.
- 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.

ExtJS lleva incluidos la mayoría de los controles de los formularios Web incluyendo Grids para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería de componentes.

#### 2.4.1.2 YUI

Yahoo! User Interface (YUI) Library es un conjunto de utilidades y controles escritos en Javascript para construir aplicaciones Web usando técnicas como DOM scripting, DHTML y AJAX. La librería YUI

también incluye en recursos CSS. Todos los componentes en la librería YUI son Open Source bajo licencia BSD y son gratuitos para todos los usuarios.

### Características

Tiene dos tipos diferentes de componentes: utilidades y controles. Las utilidades de YUI simplifican el desarrollo de soluciones válidas para diferentes navegadores, creando una capa de abstracción a base de DOM scripting y añadiendo a las aplicaciones Web un soporte DHTML y AJAX sencillo. Los controles de YUI proporcionan elementos visuales prediseñados con una alta interactividad para tus páginas Web. Estos elementos son creados y manejados totalmente en la capa cliente (browser) y no requieren que la página sea refrescada. (http://developer.yahoo.com/yui/, 2010)

Utilidades disponibles:

1. Animation**:** Crea "efectos cinematográficos" en tus páginas a base de animar la posición, el tamaño, la opacidad u otras características de los elementos de la página. Estos efectos pueden ser usados para llamar la atención de los usuarios sobre ciertas partes de la página y permitir un interfaz mucho más amigable e intuitivo.
2. Browser History Manager**:** Los desarrolladores de aplicaciones Web muchas veces necesitan tener control sobre el estado de las páginas Web, especialmente es interesante controlar el efecto que causa sobre sus aplicaciones si el usuario pulsar el botón "Atrás" de su navegador. Browser History Manager proporciona punteros a páginas (contienen todas sus variables de estado) y control sobre el botón "Atrás" del navegador para las aplicaciones Web.
3. Connection Manager**:** Está utilidad nos ayuda a gestionar los eventos XMLHttpRequest (normalmente conocido como AJAX). También incluye un completo soporte para realizar envíos de formularios (form posts), control de errores (error handling) y callbacks. Por último, Connection Manager también soporta el envío de ficheros.
4. DataSource Utility**:** Proporciona un interfaz para recibir datos desde arrays, servicios XHR y funciones personalizadas con soporte de caché e integradas con la utilidad de Connection Manager.

5. Dom Collection**:** Es una colección de objetos que permiten simplificar las tareas comunes de DOM-scripting, incluyendo el posicionamiento de elementos y la gestión del estilo CSS.
6. Drag & Drop**:** Permite crear objetos "arrastrables" ("draggable") que pueden ser cogidos y soltados en cualquier parte de la página. La utilidad permite trabajar de forma sencilla y transparente con todos los navegadores soportados.

#### 2.4.1.3 Dojo

Dojo es un framework que contiene APIs y widgets (controles) para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, los efectos de UI, drag and drop APIs, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX.

Los complementos de Dojo son componentes pre empaquetados de código Javascript, HTML y CSS que pueden ser usados para enriquecer aplicaciones web.

Una característica importante de las aplicaciones AJAX es la comunicación asíncrona entre el navegador y el servidor. Tradicionalmente, se realizaba con el comando Javascript XMLHttpRequest. Dojo provee de una capa de abstracción (dojo.io.bind) para varios navegadores web con la que se pueden usar otros transportes (como IFrames ocultos) y diferentes formatos de datos. De esta forma podemos obtener los campos que se van a enviar como parámetros del formulario de una manera sencilla. (http://www.dojotoolkit.org/, 2010).

Provee de un sistema de paquetes para facilitar el desarrollo modular. El script de inicio inicializa una serie de jerarquías de paquetes de espacios de nombre bajo el paquete raíz dojo. Después de la inicialización del paquete dojo, cualquier otro paquete puede ser cargado (vía XMLHttpRequest o cualquier otro transporte similar) usando las utilidades ofrecidas en el arranque. También es posible

inicializar paquetes adicionales dentro o al mismo nivel que el paquete dojo, permitiendo extensiones o bibliotecas de terceros.

Los paquetes de Dojo pueden contener múltiples archivos. Cualquier paquete o archivo puede depender de otro. En este caso, cuando el paquete es cargado, cualquier dependencia será también cargada.

Dojo también brinda una manera de crear perfiles; el sistema ofrece una lista de paquetes y usa Apache Ant para crear un archivo Javascript comprimido que contiene dichos paquetes y dependencias. De esta manera se tiene todo el código necesario para ser cargado y es inicializado de una sola vez, permitiendo así el cacheado (la mayoría de los navegadores web no permiten el cacheado de archivos vía XMLHttpRequest).

#### 2.4.1.4 Prototype

Prototype es un framework Javascript desarrollado por Sam Stephenson para el desarrollo sencillo y dinámico de páginas Web. Prototype nos simplifica gran parte del trabajo cuando se pretende desarrollar páginas altamente interactivas. (http://www.prototypejs.org/, 2010).

Proporciona diversas funciones para el desarrollo de aplicaciones Javascript. Su rango de funciones va desde programar accesos directos hasta avanzadas funciones de programación para tratar con XMLHttpRequest.

También proporciona funciones de librería para soportar clases y objetos basados en clase (class-based), algo que el lenguaje Javascript no tiene. En cambio en Javascript, la creación de objetos está basada en prototipos, la función de crear de un objeto puede tener una propiedad de prototipo, y cualquier objeto asignado a esa propiedad será utilizado como un prototipo para los objetos creados con esa función. El framework Prototype no debe ser confundido con esta característica del lenguaje.

El potencial de Prototype es aprovechado al máximo si se desarrolla con Ruby On Rails, esto no quiere decir que no se puede usar otro lenguaje, sólo que demandará un "mayor esfuerzo" en el desarrollo.

#### 2.4.1.5 JQuery

JQuery es una biblioteca o framework Javascript, creada inicialmente por John Resig, la cual permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a aplicaciones web. Esta librería Javascript fue presentada el 14 de enero de 2006 en el Bar Camp NYC.

Al igual que otras bibliotecas, jQuery ofrece una serie de funcionalidades basadas en Javascript que de otra manera requerirían de mucho más código ya que con las funciones propias de esta biblioteca se logran grandes resultados reduciendo en gran escala los tiempos de desarrollo de las aplicaciones. (http://jquery.com/, 2010).

**Características**:

JQuery consiste en un único fichero Javascript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery ().

Un ejemplo de las extensiones de jQuery lo constituye jQuery UI

JQuery UI es una biblioteca de componentes para el framework Javascript jQuery la cual le añade un conjunto de plugins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o

módulo de jQuery UI se desarrolla de acuerdo a la filosofía de jQuery (“find something, manipulate it” : “encuentra algo, manipúlalo”).

## 2.5 Desarrollo de componentes para estos frameworks Javascript.

La tendencia de desarrollo que siguen los frameworks Javascript, o al menos los más populares, parten del paradigma de componentes con vista a aprovechar las ventajas planteadas anteriormente. ExtJS y Dojo Toolkit son de los ejemplos más significativos en este sentido, ambos tienen comunidades muy activas que desarrollan continuamente piezas de software encapsuladas que pueden reutilizarse en otras aplicaciones.

Este trabajo no está centrado en el desarrollo de componentes para ExtJS, el cual es el framework Javascript utilizado para el desarrollo de aplicaciones en el marco del Departamento de Soluciones para la Aduana, aún cuando se explotan sus características y los mecanismos que propone para realizar este tipo de desarrollo. El alcance que se persigue está más allá de la capa de presentación de los sistemas, la cual constituye, en última instancia, el único interés de ExtJS sino que se persigue desarrollar un grupo de componentes que den respuesta a requerimientos funcionales específicos del dominio aduanal, que encapsulen en un único componente tanto la capa de presentación como la lógica de negocio asociada al mismo, todo esto sin perder de vista la necesidad de mantener lo más cercano a cero posible el acoplamiento entre las capas, de manera tal que puedan ser desensambladas y utilizadas en el desarrollo de otros componentes con altos niveles de eficiencia**.**

# CAPÍTULO 3: ANÁLISIS Y DESCRIPCIÓN DE LA SOLUCIÓN

## 3.1 Diseño de la solución

Es necesario partir de que Javascript es un lenguaje interpretado por los navegadores en el cliente, y no comprende el paradigma de la programación orientada a objetos. Esto nos lleva a que los conceptos de clase, componente y objeto, no son manejados explícitamente por este lenguaje, sino que son simulados a partir de la propiedad *prototype*. Esta propiedad permite gestionar los atributos y métodos de un objeto, brindando un comportamiento similar a una clase en la programación orientada a objetos.

El framework ExtJS maneja internamente la propiedad prototype, y propone su propia estructura para crear funciones que serán manejadas como clases, y que incluyen simuladamente el concepto de herencia en la implementación de la función *Ext.extend()*. De esta forma el framework permite crear nuevas funciones que heredan el comportamiento de otras funciones, por lo cual pueden ser denominados indistintamente los conceptos de clase, herencia, polimorfismo, atributo e instancia.

Al igual que la función *Ext.extend()*, ExtJS brinda las funciones *Ext.ns()* y *Ext.reg()*, para crear paquetes de clases y registrarlas para ser reconocidas por el framework respectivamente. Los componentes visuales a desarrollar se ajustarán a un modelo estructural de clase para su implementación, cumpliendo así el estándar propuesto por ExtJS para la creación de nuevas clases (ver figura #1).

Cada lenguaje de programación orientado a objetos promueve viablemente una manera de invocar al constructor y demás métodos de la clase base, para esto Javascript implanta el uso de la propiedad *superclass,* la cual establece las relaciones entre los métodos de las clases hijas y los de las clases bases.

Para instanciar las clases correspondientes a los componentes, ExtJS hace uso de los atributos de configuración de Javascript, los cuales argumentan los valores por defecto de los atributos de la clase,

redefiniéndoles nuevos valores y logrando que las instancias posean las características deseadas en cada caso (ver figura #2).

```
//Nombre del paquete donde se incluyen los componentes
Ext.ns("Package");

//Nombre del componente dentro del paquete
Package.NameComponent = Ext.extend(

    //Nombre de la clase padre, ejemplo: Ext.Panel, Ext.form.Field
    Ext.form.Field,

    //Cuerpo de la nueva clase
    {
        //Atributos de la clase
        attribute1: 'default value',

        //Constructor de la clase
        initComponent: function(){
            //Cuerpo del constructor

            //Inicialización del constructor en la clase padre
            Package.NameComponent.superclass.initComponent.call(this);
        },

        //Métodos de la clase con sus respectivos argumentos
        method1: function(agrs0, agrs1, agrsN){

        }
});

//Se registra el componente para ser reconocido por Ext JS
Ext.reg('XTypeComponent', Package.NameComponent);
```

Figura 1 Modelo de clase para implementar los componentes visuales.

```
var instancia1 = new Package.NameComponent({
    attribute1: 'Valor 1'
});

var instancia2 = new Package.NameComponent({
    attribute1: 'Valor 2'
});
```

Figura 2 Ejemplos de creación de instancias de los componentes.

## 3.2 Análisis e identificación de los componentes visuales

Para la identificación de estos componentes se realizó un análisis de los requerimientos de interfaz, los cuales en conjunto con el negocio planteado para el sistema aduanal, coincide en menor o mayor medida con varios casos de usos de diferentes módulos. Al cumplir estos requerimientos de interfaz con tal grado de similitud, se pueden agrupar las características y funcionalidades comunes para ser comprendidas en componentes, y dar la posibilidad de ser configurables y reutilizables.

Anteriormente se han desarrollado estas interfaces de tal modo que no permiten reutilizar el código de las mismas, sino replicarlo una y otra vez en la medida que se necesite. Por tal motivo se han numerado estas interfaces y analizado sus comportamientos, para encapsular en clases tanto los requerimientos visuales como parte de la lógica del negocio asociada, evitando así que el desarrollador implemente una vez más la interfaz y las funcionalidades que esta brinda.

Tras desarrollado el análisis de los requerimientos de interfaz, se identificaron un conjunto de componentes visuales por la cantidad de apariciones que estos hacen en los casos de usos estudiados hasta el momento (ver tabla # 2).

Es importante destacar que el desarrollo de estos componentes está basado en el algoritmo de planificación MFU - (*more frequently used*) -, el más frecuentemente usado. Este algoritmo plantea que la mayor prioridad recae sobre el elemento que mayor probabilidad de ocurrencia tiene, de esta forma se desarrollarían los componentes de mayor frecuencia en los casos de usos estudiados, y los tiempos de desarrollo de las interfaces correspondientes a estos casos de usos serían menores.

En el caso del componente Viewport, se determinó que tenía una prioridad superior a los otros, ya que controla toda la capa de presentación principal de cada módulo del sistema, y constituye el escenario donde actúan las interfaces que contienen parcial o totalmente alguno de los demás componentes.

| Requerimientos de Interfaz | Componente | Ocurrencias |
|---|---|---|
| ComboBox que carga datos de una determinada tabla de la base de datos. | TcCombo | 104 |
| Interfaz para agrupar funcionalidades de formularios y distribución de elementos en columnas. | ColumnPanel, ColumnFieldset | 87 |
| Mecanismo para guardar información en documentos generados con formato digital PDF. | XExporter | 48 |
| Interfaz para gestionar un formulario con las funcionalidades de consultar, adicionar, modificar y eliminar datos. | FormGridPanel | 43 |
| Interfax para realizar búsquedas dado los criterios necesarios. | SearchPanel | 41 |
| Interfaz para realizar transferencias de ficheros entre el ordenador cliente y el servidor. | UpLoadFile | 37 |
| Interfaz para gestionar las fotos de las personas utilizando las funcionalidades de consultar, adicionar, modificar y | PhotoPanel | 33 |

| | | |
|---|---|---|
| eliminar imagen. | | |
| ComboBox que carga sus datos de una tabla de la base de datos, referencia a otro ComboBox y recarga sus datos de acuerdo a un criterio de selección. | TcRefCombo | 24 |
| Interfaz para intercambiar información entre dos tablas. | GridToGridPanel | 12 |

Tabla 2 Ocurrencia de los requerimientos de interfaz identificados como componentes.

## 3.3 Descripción de la solución

Teniendo en cuenta la arquitectura de dominio específico del GINA, se realizó el desarrollo de los componentes aplicando los estándares definidos. El conjunto de componentes en cuestión, establece un margen a seguir por los desarrolladores, lo cual facilita que estos se acojan a dicho estándar y les sea más comprensible a futuros desarrolladores para su optimización y modificación.

Cada uno de estos componentes comprende parte de la lógica del negocio, ya sea en el servidor o en el cliente. Incluyen desde validaciones hasta captura de errores enviadas por el servidor, interactúan con el usuario del sistema a través de mensajes informativos, de alerta, de errores y de preguntas, utilizan botones que ejecutan acciones asociadas a la clase, y muestran descriptores de herramientas en elementos determinados para su mejor entendimiento.

Garantizan el trabajo con eventos, basado en el propio lenguaje de programación Javascript y en un mecanismo implementado por el framework ExtJS. Al igual que en cualquier lenguaje de programación orientado a objetos, los eventos se apoyan en parámetros y permiten ejecutar funciones creadas por los desarrolladores, utilizando los parámetros correspondientes a cada evento para realizar la acción deseada.

Para una mejor comprensión del desarrollo de los componentes se describen sus funcionalidades y características en los siguientes apéndices:

### 3.3.1 Viewport

Es el componente principal de la interfaz del sistema, ya que controla la capa de presentación de cada módulo, ocupa todo el cuerpo de la página en el navegador y se compone de un conjunto de elementos que brindan información y funcionalidades al usuario del sistema. Estos elementos se encuentran agrupados en dos regiones fundamentales del componente creando un marco visual estándar para cada uno de los subsistemas de la aplicación.

Elementos que definen al componente por regiones:

1. Región norte.
    - Nombre del usuario autenticado en el sistema.
    - La hora y fecha del ordenador cliente.
    - Imagen que identifica al subsistema.
    - Vínculo para acceder a terminar la sesión del usuario autenticado.
2. Región central.
    - Menú que carga las funcionalidades a la cual el usuario autenticado tiene acceso. La sesión de
    - Contenedor donde se muestran las interfaces de los casos de usos del módulo.

| Viewport::Ext.Viewport | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | bannerCls | Nombre del selector de clase CSS que especifica una imagen de fondo a ser usada como logotipo del módulo. |

| | | |
|---|---|---|
| | menu | Listado de elementos que constituyen el menú del módulo. |
| **Métodos** | logoff | Termina la sesión del usuario autenticado. |

Tabla 3 Descripción del componente Viewport

### 3.2.2 TcCombo

El componente TcCombo no es más que una especialización de *Ext.form.ComboBox*, por lo que hereda las mismas características como elemento de formulario y componente visual. Por defecto la clase base define valores a sus atributos que no se corresponden en su totalidad con el estándar a cumplir para este tipo de elemento, para dar solución a la necesidad de aplicar dicho estándar, el componente TcCombo redefine los valores de los atributos de la clase, y define nuevos atributos a utilizar en el funcionamiento de la clase.

Su utilidad está orientada a cargar los datos de una determinada tabla de la base de datos, para esto implementa un mecanismo dinámico que permite dado el nombre físico de la tabla, obtener la descripción e identificador de cada una de las unidades de datos persistentes en la tabla, basándose en los estándares definidos tanto en la base de datos como en las clases del negocio que interactúan con ella.

Debido al dinamismo que implementa el componente, las consultas se centralizan en una única función, y evita que los desarrolladores implementen repetidamente el mecanismo empleado para cada una de las tablas de la base de datos. Además genera el atributo que almacena los datos una vez realizada la petición al servidor, aunque permite que se configuren todas sus propiedades y se utilice como un ComboBox normal, cambiando su política de funcionamiento pero manteniendo el estándar.

**TcCombo::Ext.form.ComboBox**

| Elementos de la clase | Nombre | Descripción |
|---|---|---|
| **Atributos de configuración** | autoLoad | Verdadero si se desea que se carguen los datos desde que se crea el objeto. |
| | tabla | Nombre físico de la tabla de la base de datos de la cual se toman los datos. |

Tabla 4 Descripción del componente TcCombo

### 3.2.3 TcRefCombo

El TcRefCombo es una especialización del componente TcCombo, adopta el mismo comportamiento y adiciona una nueva funcionalidad fundamental, orientada a solucionar las relaciones de dependencia entre dos tablas de la base de datos. La relación comprendida por las tablas es de tipo uno a muchos, es decir, un elemento de una tabla puede aparecer en varios elementos de la otra.

Considerando que la relación se establece entre dos tablas de la base de datos, para realizar este vínculo de forma visual, se necesita que cada instancia del TcRefCombo se relacione con otro objeto de igual parecido, ya sea de tipo ComboBox o alguna especialización de este. La relación entre ambos objetos se logra a través de una referencia, la cual constituye el identificador del objeto referenciado, de esta forma la clase puede obtener en cualquier momento el objeto de referencia utilizando su identificador, y acceder a sus métodos y atributos.

Su funcionalidad básica consiste en que cada vez que se selecciona un elemento de la lista desplegable, se recargan automáticamente los datos del ComboBox referenciado pasando como parámetro el identificador del elemento seleccionado. Con esto se logra que al recargar los datos del objeto de referencia, tome en cuenta el parámetro enviado, lo cual significa que los nuevos datos se relacionan con el valor del parámetro.

Es importante señalar que inicialmente el TcRefCombo deshabilita el objeto al que hace referencia, ya que es necesario tener una base para cargar sus datos. Además, al seleccionar un elemento, primero se restablece el objeto referenciado a su estado inicial, luego se recargan sus datos y se habilita, en caso de ser nulos los datos recargados, el objeto permanece deshabilitado y se muestra el mensaje correspondiente.

Un ejemplo de la utilidad de este componente y cómo funciona es el caso de las provincias y municipios de Cuba, donde cada provincia tiene varios municipios. En la base de datos provincia y municipios serían dos tablas donde la tabla municipio hace referencia a la tabla provincia, es decir cada municipio tiene una

referencia al identificador de la provincia que pertenece. En la interfaz un objeto TcRefCombo cargaría sus datos de la tabla provincia e hiciera referencia al identificador de otro objeto TcCombo que carga sus datos de la tabla municipio. Cada vez que se seleccione una provincia, se recargarán automáticamente los datos de los municipios correspondientes a la provincia seleccionada (ver figura # 3).

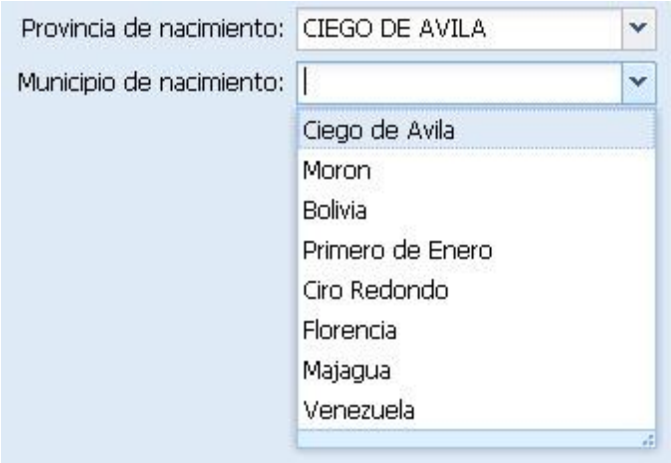


Figura 3 Ejemplo de utilización del TcRefCombo.

| TcRefCombo::TcCombo | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | idNextCombo | Nombre del identificador del elemento de referencia. |
| **Métodos** | restoreNext | Restablece los valores del elemento de referencia a su estado inicial y lo deshabilita en tanto no se carguen sus datos nuevamente. |
| **Eventos** | reloaded | Se dispara en cuanto terminan de cargarse los datos del elemento |

| | | |
|---|---|---|
| | | referenciado. |

Tabla 5 Descripción del componente TcRefCombo

### 3.2.4 ColumnPanel y ColumnFieldset

Ambos presentan el mismo comportamiento, características y funcionalidades, diferenciados únicamente por adquirir la herencia de distintas clases bases, *Ext.Panel* para el caso del ColumnPanel y *Ext.form.Fieldset* para el ColumnFieldset. Desde el punto de vista visual, estas clases bases poseen amplias diferencias, pero mantienen un comportamiento similar como elementos contenedores, por lo que brindan las mismas funcionalidades permitiendo que se implemente un único código para ambas clases, y que comprendan a dos componentes con características visuales diferentes e igual funcionalidad.

Facilitan la distribución de elementos en columnas y permiten agrupar las características comunes de sus elementos. Prestan un conjunto de funcionalidades que sirven como herramientas para el trabajo con los elementos de formulario, y posibilitan obtener sus valores, modificarlos, validarlos, restablecer los valores a su estado inicial y adicionar nuevos elementos.

Alcanzan su mayor potencial en la generación dinámica de formularios, que tiene su base en atributos de configuración que controlan diferentes formas de distribución de elementos, atributos que establecen la cantidad y tamaño de las columnas, así como la correspondencia entre los elementos a distribuir y las columnas previamente generadas.

Aunque crean las condiciones para el uso directo de elementos de formulario, no se limitan a este ámbito sino que aceptan cualquier componente visual y lo distribuye indistintamente del tipo que sea. En el caso de que los elementos inmediatos sean contenedores de elementos de formulario, se mantienen activas las herramientas de formulario y en su correcto funcionamiento.

Debido al gran dinamismo que presentan para generar formularios y mantener la política de configuración establecida por el framework ExtJS, ambos componentes se incluyen en el desarrollo de los componentes FormGridPanel y SearchPanel, los cuales explotan sus funcionalidades y reutilizan sus atributos de configuración.

| ColumnPanel::Ext.Panel, ColumnFieldSet::Ext.form.FieldSet | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | columns | Determina la cantidad de columnas y sus dimensiones en dependencia de la configuración dada:<br>➢ Por defecto: se genera una columna para cada elemento.<br>➢ Entero natural: se genera el número de columnas especificado.<br>➢ Lista numérica: se generan tantas columnas como elementos tenga la lista, especificando el ancho de cada columna según el número correspondiente, si el número es mayor que uno se toma en píxeles, sino en por ciento. |
| | itemsConfig | Se definen las configuraciones que son comunes para todos los elementos. |
| | labelAlign | Define la alineación de las etiquetas de los campos de formulario en las columnas. |
| | defaultType | Define el tipo de elemento por defecto a utilizar en las columnas. |
| | vertical | Determina la forma de distribuir los elementos por las columnas en dependencia de la configuración dada:<br>➢ Falso: se distribuyen los elementos uno por uno sucesivamente por cada |

| | | |
|---|---|---|
| | | columna.<br>➢ Verdadero: se distribuyen verticalmente los elementos en cada columna de tal forma que las de la izquierda posean mayor o igual cantidad de elementos que las columnas de la derecha. |
| | columnIndex | Verdadero para distribuir los elementos por las columnas teniendo en cuenta el índice especificado por cada elemento. Se obvia la propiedad vertical. |
| | labelWidth | Define el ancho de las etiquetas de los campos de formulario en las columnas. |
| **Métodos** | getFields | Retorna un listado con todos los elementos de formulario. |
| | reset | Restablece los valores de todos los elementos de formulario a su estado inicial. |
| | isValid | Valida todos los campos de formulario y retorna verdadero si todos son válidos, falso en caso contrario. |
| | getValues | Retorna los valores de los campos de formulario. |
| | loadRecord | Carga en los campos de formulario un registro de datos dado como parámetro. |
| | addItem | Adiciona un nuevo elemento en la columna especificada. |
| **Eventos** | valid | Se produce luego de haberse ejecutado la acción del método isValid. |

| | | |
|---|---|---|
| | loadRecord | Se produce luego de haberse ejecutado la acción del método loadRecord. |

Tabla 6 Descripción del componente ColumnPanel y ColumnFieldset

### 3.2.5 FormGridPanel

Panel que divide su área en dos regiones fundamentales, una región para el formulario y otra para la tabla donde se registran los datos. Para la inclusión del formulario el componente hace uso del ColumnPanel y ColumnFieldset, reutilizando sus propiedades generativas y dinámicas, aunque permite que el desarrollador defina su propio formulario en caso conveniente. El formulario se determina a través de atributos de configuración, que definen cuando utilizar el componente ColumnPanel o el ColumnFieldset, y cuando el desarrollador ha definido el formulario a visualizar.

El componente utiliza el formulario determinado para generar la tabla asociada a los elementos de formulario que lo componen. Para lograr esta asociación se tiene en cuenta algunos de los parámetros configurados en cada uno de los campos del formulario, parámetros que determinan qué campo está asociado a qué columna de la tabla, si se genera la columna que corresponde al campo, y en ese caso si inicialmente permanece oculta.

El formulario generado o definido por el desarrollador puede ser posicionado en cualquiera de las regiones norte, sur, este u oeste, por otro lado la tabla siempre aparece situada en la región central, y a pesar de ser generada, el desarrollador tiene acceso a adicionar y redefinir propiedades, alcanzando en ambos casos la máxima configuración deseada.

Este tipo de componente es muy útil cuando se desea llenar varias veces un mismo formulario y almacenar los datos temporalmente en una tabla, para ello el componente implementa las funcionalidades de adicionar, modificar y eliminar. Además soporta propiedades que permiten arrastrar y

soltar elementos, en este caso sería arrastrar una fila de la tabla sobre el formulario para acceder cómodamente a la opción modificar.

El funcionamiento del componente como un todo se logra gracias al apoyo de las funciones de la clase, que pretenden visual y funcionalmente ejecutar las siguientes actividades:

- Mostrar los mensajes de errores correspondientes a la validación del formulario o la selección de elementos de la tabla para su modificación o eliminación.
- Cargar los datos de una fila de la tabla en los campos del formulario.
- Obtener los valores almacenados en la tabla para ser enviados o tratados como objetos.
- Obtener los valores de los campos de formularios previamente validados y adicionarlos como una fila en la tabla.
- Actualizar los datos de los campos modificados en la fila correspondiente de la tabla.
- Eliminar los datos de las filas seleccionadas en la tabla.
- Limpiar los campos del formulario para ser ingresados nuevamente los datos.

| FormGridPanel:: Ext.Panel | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | formRegion | Define la región donde se ubicará el formulario. |
| | formType | Define el tipo de formulario a utilizar, ColumnPanel o ColumnFieldset. |
| | gridTitle | Establece el título de la tabla generada. |
| | form | Formulario definido por el usuario. |

| | | |
|---|---|---|
| | loadURL | Dirección donde se cargan inicialmente los datos de la tabla generada. |
| | formSize | Ancho o alto del formulario en dependencia de la región en que se encuentre. |
| | fields | Listado de campos para generar el formulario. |
| | fieldsConfig | Configuraciones comunes para todos los campos del formulario. |
| | gridConfig | Configuraciones a ser aplicadas a la tabla generada. |
| | formConfig | Configuraciones a ser aplicadas al formulario generado. |
| **Métodos** | isVertical | Retorna verdadero si el formulario se encuentra en las regiones norte o sur, en caso contrario retorna falso. |
| | getForm | Retorna el formulario. |
| | getGrid | Retorna la tabla generada. |
| | isEditing | Retorna verdadero si se está editando alguna fila de la tabla, en otro caso retorna falso. |
| | getFields | Retorna un listado con los elementos del formulario. |
| | reset | Restablece los valores de todos los elementos de formulario a su estado inicial. |
| | isValid | Valida todos los campos de formulario y retorna verdadero si todos son válidos, falso en caso contrario. |
| | getValues | Retorna los valores de los campos de formulario. |
| | loadRecord | Carga en los campos de formulario un registro de datos dado como parámetro. |
| | getStoreValues | Retorna los datos almacenados en la tabla. |
| | showInvalidMsg | Muestra un mensaje de error que indica la existencia de datos inválidos en los campos del formulario. |
| | showEmptySelectionMsg | Muestra un mensaje de error que indica la carencia de selección de alguna fila de la tabla para realizar las acciones |

| | | |
|---|---|---|
| | | de eliminar o modificar. |
| | changeState | Habilita o deshabilita el área de la tabla y las funcionalidades de adicionar y eliminar, en dependencia de si se está editando o no. |
| | doAdd | Procede a adicionar los datos de los campos del formulario en una nueva fila de la tabla. Valida previamente los campos del formulario, en caso de determinar al menos un campo inválido, se muestra el mensaje de error correspondiente y se cancela la operación. |
| | doEdit | Procede a cargar los datos de una fila de la tabla en el formulario para ser modificados, y luego salvar los cambios realizados. |
| | doDelete | Procede a eliminar de la tabla las filas seleccionadas. En caso de no existir selección en la tabla, se muestra el mensaje de error correspondiente. |
| | disableValidations | Guarda la validación de cada campo y la modifica de tal forma que siempre retorna verdadero. |
| | restoreValidations | Restablece en cada campo la validación guardada. |
| **Eventos** | beforeAdd | Se produce antes de realizarse la acción de adicionar. En caso de retornarse falso, se detiene la acción adicionar. |
| | beforeEdit | Se produce antes de realizarse la acción de modificar. En caso de retornarse falso, se detiene la acción modificar. |
| | beforeDelete | Se produce antes de realizarse la acción de eliminar. En caso de retornarse falso, se detiene la acción eliminar. |
| | add | Se produce una vez terminada la acción adicionar. |
| | edit | Se produce una vez terminada la acción modificar. |
| | delete | Se produce una vez terminada la acción adicionar. |

| | | |
|---|---|---|
| | loadRecord | Se produce luego de haberse ejecutado la acción del método loadRecord. |
| | valid | Se produce luego de haberse ejecutado la acción del método isValid. |

Tabla 7 Descripción del componente FormGridPanel

### 3.2.6 SearchPanel

El SearchPanel es un componente orientado a facilitar el desarrollo de las interfaces de búsquedas. Cómo toda búsqueda se limita a un grupo de criterios que marcan el margen por el cual se rige su resultado. Para el SearchPanel los criterios están dados por elementos de formulario definidos por el desarrollador en un formulario propio o generado a través de los componentes ColumnPanel y ColumnFieldset.

Se incluye por defecto los criterios Fecha Inicial y Fecha Final, debido a que un gran número de las búsquedas registradas en los casos de uso especifican su utilidad, en caso contrario el desarrollador puede deshabilitar esta opción. De ser utilizados ambos criterios, se implementa un tipo de validación especial para estos, que consiste en imposibilitar que el usuario una vez seleccionada una fecha inicial acceda a seleccionar una fecha final inferior, y viceversa. Esta validación confirma que el rango de fecha sea en todo momento válido, y evita las continuas validaciones en el servidor.

El comportamiento general de este componente consiste en enviar al servidor los datos de los campos de formulario previamente validados, los cuales constituyen los criterios de la búsqueda a realizar, y mostrar el resultado de la petición en la tabla definida por el desarrollador. Para la realización de la búsqueda, se genera un botón que ejecuta la validación de los campos y la acción de buscar en una dirección especificada.

| SearchPanel:: Ext.Panel | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | formRegion | Define la región donde se ubicará el formulario. |
| | formType | Define el tipo de formulario a utilizar, ColumnPanel o ColumnFieldset. |
| | defaultFields | Define si se muestran o no los criterios Fecha Inicial y Fecha Final. |
| | form | Formulario definido por el usuario. |

| | | |
|---|---|---|
| | url | Dirección donde se cargan los datos de la tabla y se realiza la búsqueda. |
| | formSize | Ancho o alto del formulario en dependencia de la región en que se encuentre. |
| | fields | Listado de campos para generar el formulario. |
| | fieldsConfig | Configuraciones comunes para todos los campos del formulario. |
| | gridConfig | Configuraciones a ser aplicadas a la tabla que define el desarrollador. |
| | formConfig | Configuraciones a ser aplicadas al formulario generado. |
| **Métodos** | isVertical | Retorna verdadero si el formulario se encuentra en las regiones norte o sur, en caso contrario retorna falso. |
| | getForm | Retorna el formulario. |
| | getGrid | Retorna la tabla definida. |
| | getFields | Retorna un listado con los elementos del formulario. |
| | reset | Restablece los valores de todos los elementos de formulario a su estado inicial. |
| | isValid | Valida todos los campos de formulario y retorna verdadero si todos son válidos, falso en caso contrario. |
| | getValues | Retorna los valores de los campos de formulario. |
| | getStoreValues | Retorna los datos almacenados en la tabla. |
| | showInvalidMsg | Muestra un mensaje de error que indica la existencia de datos inválidos en los campos del formulario. |
| | disableValidations | Guarda la validación de cada campo y la modifica de tal forma que siempre retorna verdadero. |
| | restoreValidations | Restablece en cada campo la validación guardada. |
| **Eventos** | valid | Se produce luego de haberse ejecutado la acción del método isValid. |
| | find | Se produce antes de realizarse la acción de buscar. En caso de |

| | | |
|---|---|---|
| | | retornar falso no se procede a la búsqueda. |

Tabla 8 Descripción del componente SearchPanel

### 3.2.7 UploadFile

El UploadFile es una herramienta visual cuya función principal es realizar transferencias de ficheros entre el ordenador cliente y el servidor. La transferencia se realiza enviando una referencia del fichero seleccionado por el usuario, la cual tiene lugar inmediatamente después de realizada la selección, y que se basa en elementos de formulario que facilitan este tipo de trabajo.

Posee un área de información donde se muestra el estado de la operación, y cuando ocurre algún error durante la transferencia, se le informa al usuario y se cancela la acción. Generalmente los errores que se presentan son por problemas de permisos de lectura en el cliente o la falta de conexión al servidor.

Este componente implementa el patrón de diseño denominado Singleton, o lo que es lo mismo, instancia única. Este patrón impide que se utilice más de un objeto de una clase en un mismo instante de tiempo. De esta forma el UploadFile se convierte en un objeto, que puede ser referenciado de manera global y que permite modificar sus atributos públicos en lugar de crear nuevas instancias.

| UploadFile::Ext.Window | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | title | Título de la ventana del componente. |
| | iconCls | Nombre del selector de clase CSS que especifica una imagen a ser usada como ícono de la ventana del componente. |
| | url | Dirección a donde se envía la referencia del fichero seleccionado para realizar la transferencia. |
| | dir | Dirección donde se guarda fichero luego de realizar la transferencia. |
| | fn | Función a ejecutar cada vez que se realice una transferencia de forma correcta. |
| **Métodos** | show | Muestra la ventana que contiene todo el cuerpo del componente. |

Tabla 9 Descripción del componente UploadFile

### 3.2.8 PhotoPanel

El componente PhotoPanel es un elemento de formulario creado gracias al mecanismo de herencia que brinda el framework ExtJS. Toma su herencia del componente *Ext.form.Field*, lo que le permite comportarse como un elemento más de formulario, aunque visualmente es más semejante a un panel con etiquetas de imagen y botones.

Cómo todo elemento de formulario garantiza modificar y obtener sus valores, en este caso la localización de la imagen que se muestra en parte del cuerpo del PhotoPanel. Posee dos botones que presentan las funcionalidades de adicionar y eliminar imagen, definidos por defecto para modificar de forma visual el elemento, de no ser así se puede deshabilitar esta opción configurando la propiedad correspondiente.

La utilidad de este componente está enfocada en lograr un rápido y efectivo trabajo con imágenes, hasta el momento se utiliza para mostrar y cargar las fotos de las distintas personas que interactúan directa o indirectamente con el sistema. Para cargar las imágenes hace uso del componente UploadFile que le permite transferir la imagen seleccionada al servidor y luego mostrarla en el PhotoPanel.

| PhotoPanel::Ext.form.Field | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | showButtons | Define si se muestran o no los botones de adicionar y eliminar imagen. |
| | tooltipAdd | Descripción visual del botón Adicionar. |
| | tooltipDelete | Descripción visual del botón Eliminar. |
| | name | Nombre con el que se envía al servidor el valor del campo. |
| **Métodos** | reset | Restablece el valor del campo a su estado inicial. |
| | setValue | Modifica el valor del campo dado otro valor obtenido como parámetro. |

| | | |
|---|---|---|
| | getValue | Retorna el valor del campo. |
| **Eventos** | change | Se dispara cuando cambia el valor del campo. |

Tabla 10 Descripción del componente PhotoPanel

### 3.2.9 GridToGridPanel

Componente que se caracteriza por intercambiar datos entre dos tablas que poseen columnas coincidentes entre ambas tablas. Se compone de tres regiones, una para cada tabla y otra donde se ubican los botones que realizan las funciones de mover las filas seleccionadas a la otra tabla o moverlas todas. Para el caso de ser seleccionadas las filas, se puede utilizar además la propiedad de arrastrar y soltar elementos, causando el mismo efecto.

El GridToGridPanel incluye atributos de configuración a ser aplicados a cada una de las tablas por separado y agrupar sus características comunes. Debido a que el trabajo con los datos se realiza en la interfaz del ordenador cliente, se pueden obtener los datos de cada tabla para ser actualizados en el servidor.

Ambas tablas definen su propio almacén de datos y las columnas correspondientes, pero la transferencia se establece entre los datos que coinciden sus identificadores en las dos tablas. De esta manera se puede perder información al realizar las operaciones si no se corresponden todos los campos definidos en el almacén de datos.

Este tipo de componente en el negocio aduanal es muy útil cuando se desea asignar o liberar determinado medio u otro tipo de elemento, referente a algún individuo, entidad u operación. Un ejemplo constituye planificar candidatos para una prueba psicométrica, donde el usuario del sistema selecciona del listado de candidatos pendientes y los mueve al listado de candidatos planificados.

| GridToGridPanel::Ext.Panel | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | vertical | Define si se muestran las tablas una sobre la otra o a un lado. |
| | firstGridConfig | Configuración a aplicar a la primera tabla. |
| | secondGridConfig | Configuración a aplicar a la segunda tabla. |
| **Métodos** | getFirstGrid | Retorna la primera tabla. |
| | getSecondGrid | Retorna la segunda tabla. |
| | getStoreValues | Retorna los datos almacenados en ambas tablas. |
| | addRows | Mueve los datos seleccionados de una tabla a la otra. |
| | addAll | Mueve todos los datos de una tabla a la otra. |
| **Eventos** | beforeMove | Se produce antes de realizar algún movimiento. En caso de retornar falso no se procede a mover los datos. |
| | move | Se produce luego de terminada la acción de mover datos. |

Tabla 11 Descripción del componente GridToGridPanel

### 2.3.10 XExporter

Es el componente encargado de replicar los datos de las interfaces o de alguna dirección y salvarlos en formato digital. Se mescla con el funcionamiento del DOM PDF, aditamento del framework de PHP Symfony utilizado por el sistema del GINA. DOM PDF es una librería que contiene múltiples clases que a su vez, brindan todo tipo de funcionalidades para trabajar en la creación dinámica de ficheros con extensión PDF.

El XExporter implementa un grupo de plantillas que conforman en conjunto con los datos, el cuerpo del documento a generar. Estas plantillas hasta el momento se ajustan a los formularios, tablas, encabezados y pie de páginas, utilizando atributos para configurar los títulos, subtítulos, formato de fecha, número de página, logotipo y elementos de la vista previa.

Posee dos modos de funcionamiento, que definen cuando los datos se toman de una dirección especificada, o si se toman de una interfaz. Por defecto se obtienen los datos de una interfaz, para ello se implementa un método que identifica elementos de formulario y datos de tablas a partir del identificador de un objeto contenedor. De esta forma se garantiza que no se pierdan los datos sin importar el tipo de elemento contenedor.

El XExporter presenta antes de generar el fichero un modelo del documento en una vista previa, la cual visualiza la estructura y formato de los datos semejante a como se generan, permitiendo que el usuario sepa de antemano cuáles son los datos que se van a guardar, y puede restringir o ampliar el conjunto de datos en dependencia de la configuración de cada instancia de la clase.

Para determinar los datos a tomar de las interfaces se definen propiedades que limitan ciertos comportamientos de los datos presentes en formularios y tablas. A continuación se secuencian los diferentes datos que se consideran en la selección:

- Datos de elementos ocultos de formulario.

- Datos de las columnas ocultas de las tablas.
- Datos filtrados de las tablas.
- Todos los datos de formulario.
- Todos los datos de las tablas

| ` | | |
|---|---|---|
| **Elementos de la clase** | **Nombre** | **Descripción** |
| **Atributos de configuración** | includeFormData | Determina si se incluyen o no los datos de formulario. |
| | includeGridData | Determina si se incluyen o no los datos de las tablas. |
| | includeHiddenFormData | Determina si se incluyen o no los campo ocultos de formulario. |
| | includeHiddenGridData | Determina si se incluyen o no los datos ocultos o filtrados de las tablas. |
| | includeHiddenGridColumns | Determina si se incluyen o no los datos de las columnas ocultas de las tablas. |
| | showLogo | Determina si se muestra o no el logotipo. |
| | showDate | Determina si se muestra o no la fecha. |
| | dateFormat | Define el formato de la fecha. |
| | logoURL | Define la dirección del logotipo. |
| | title | Título a mostrar en las páginas del documento. |
| | subtitle | Subtítulo a mostrar en las páginas del documento. |

| | | |
|---|---|---|
| | footer | Texto a mostrar en el pie de las páginas del documento. |
| | pageNumText | Texto a mostrar junto al número de la página. |
| | containerId | Nombre del identificador del componente contenedor de la información a generar en el documento. |
| | viewURL | Dirección de donde se carga la vista con los datos a generar en el documento. |
| | formDataText | Texto a mostrar para identificar los datos de formulario. |
| | mode | Define el modo a emplear para generar el documento, si se toman los datos de un contenedor o de una vista predefinida. |
| **Métodos** | isContainer | Determina si un elemento es un contenedor o no. |
| | isGrid | Determina si un elemento es una tabla o no. |
| | isField | Determina si un elemento es un campo de formulario o no. |
| | showPreview | Muestra una ventana con la vista previa de los datos a generar en el documento. |
| | getValues | Retorna todos los datos de formulario y de tablas encontrados si el modo de generación es a través de un elemento contenedor. En caso contrario retorna vacío. |
| **Eventos** | export | Se produce inmediatamente terminado de generarse el documento correctamente. |

Tabla 12 Descripción del componente XExporter

# CAPÍTULO 4: FACTIBILIDAD DE LA SOLUCIÓN

## 4.1 Caso de estudio

La pertinencia de la solución propuesta es demostrada a partir del desarrollo de un caso de estudio donde se desarrollan las funcionalidades del caso del caso de uso Crear Ficha del módulo Selección perteneciente al subsistema de Recursos Humanos. El mismo fue implementado tanto desde los componentes desarrollados en este trabajo como con el mecanismo de implementación previo a los mismos. Los resultados alcanzados son suficientes para avalar el cumplimiento de los objetivos de esta investigación.

Desde que se comenzó el proceso de implementación del módulo Selección en enero del 2008 y hasta la actualidad, este caso de uso de se ha desarrollado tres veces. Las nuevas versiones del caso de uso muestran a su medida un incremento en la eficiencia de diferentes parámetros que establecen la integridad del proceso de desarrollo de la interfaz en general.

La inexperiencia en el trabajo con el framework ExtJS y la falta de un estándar produjo que la primera versión de la interfaz del caso de uso tomara un prolongado tiempo de implementación (ver figura # 4). El código presentado se caracterizaba por ser extenso y prácticamente ilegible, lo que trajo como consecuencia que en ocasiones hasta el propio desarrollador desconociera la ubicación de las funcionalidades en el código o que otros desarrolladores esperaran por la presencia del autor para realizar alguna operación de cambio.

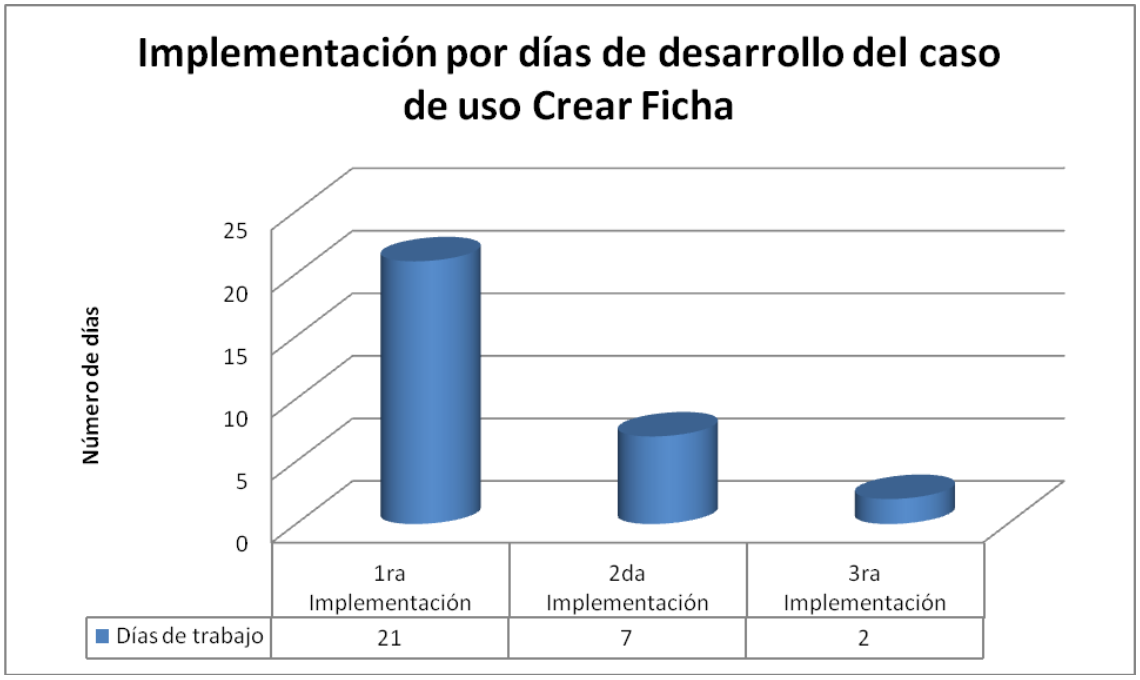


Figura 4 Desarrollo de las implementaciones del caso de uso Crear Ficha.

Los nuevos desarrolladores carecían de una capacitación básica que les permitiera junto a los problemas planteados ejecutar algún tipo de mantenimiento entre la inmensa cantidad de líneas de código abarcadas por la implementación y la fragmentación del mismo en varios ficheros (ver figura # 5), asignándose esta tarea al responsable de la implementación de la interfaz, el cual para determinar la ubicación de las funciones dentro del código, modificarlas y probar sus correctos funcionamientos empleaba un tiempo relativamente largo.

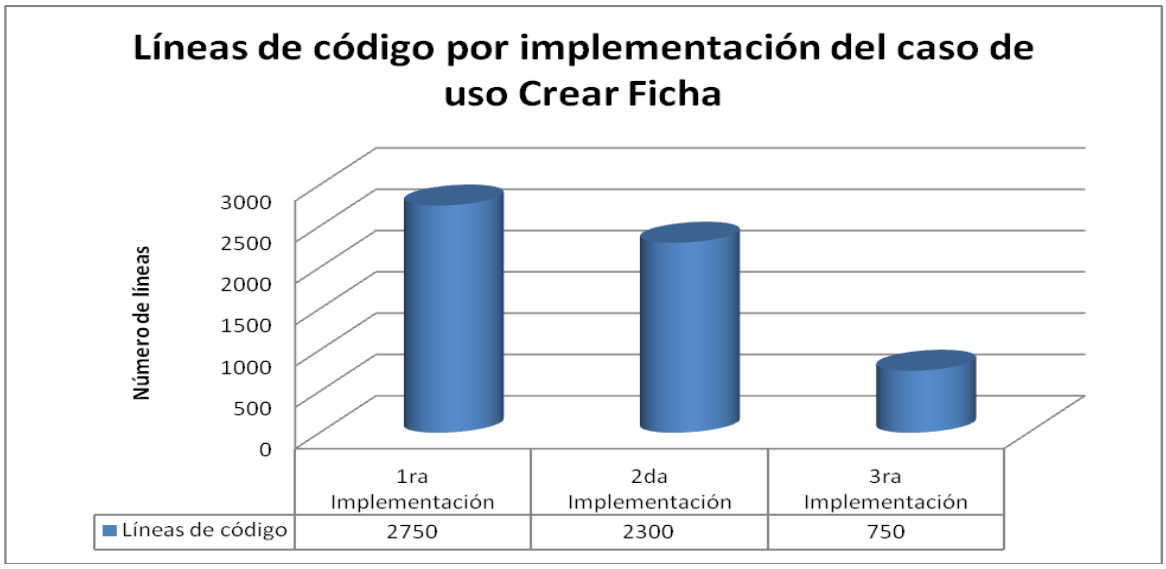


Figura 5 Cantidades de líneas de código de las implementaciones del caso de uso Crear Ficha.

Luego de definirse un estándar que comprendía los términos más generales para la capa de interfaz de usuario, se decidió crear una vez más la interfaz de este caso de uso aplicando el nuevo estándar. En este caso se mejoró en cuanto a legibilidad y comprensión del código, la demora de la creación resultó ser inferior a la anterior pero tardía y seguían presentándose problemas para lograr la estabilidad del funcionamiento y el mantenimiento del caso de uso. Esto se debía a que aun presentaba un gran tamaño de líneas de código en la implementación (ver figura # 5), ya que se repetía reiteradas veces el mismo código de los atributos de configuración de determinados objetos y funcionalidades de la misma índole.

Por todos los problemas presentados en las anteriores implementaciones, se realizó un análisis de los requerimientos de interfaz de la mayor cantidad de casos de uso descritos, para identificar un conjunto de comportamientos comunes y encapsularlos en componentes, lo cual constituye uno de los objetivos específicos de este trabajo.

El desarrollo de los componentes identificados obligó a especificar algunos términos en el estándar definido e incorporar otros nuevos, de este modo se llevó a cabo la tercera implementación del caso de uso Crear Ficha, esta vez utilizando los componentes desarrollados. Esta tercera versión marcó récord en los parámetros a medir en el proceso de desarrollo del caso se uso, se terminó la implementación en el menor tiempo registrado, se redujo considerablemente la cantidad de líneas de código, gran parte de la estabilidad y el mantenimiento están asociados al uso de los componentes, y la velocidad de carga del fichero correspondiente al caso de uso se incrementó en la medida que decreció el tamaño del fichero (ver figura # 6).

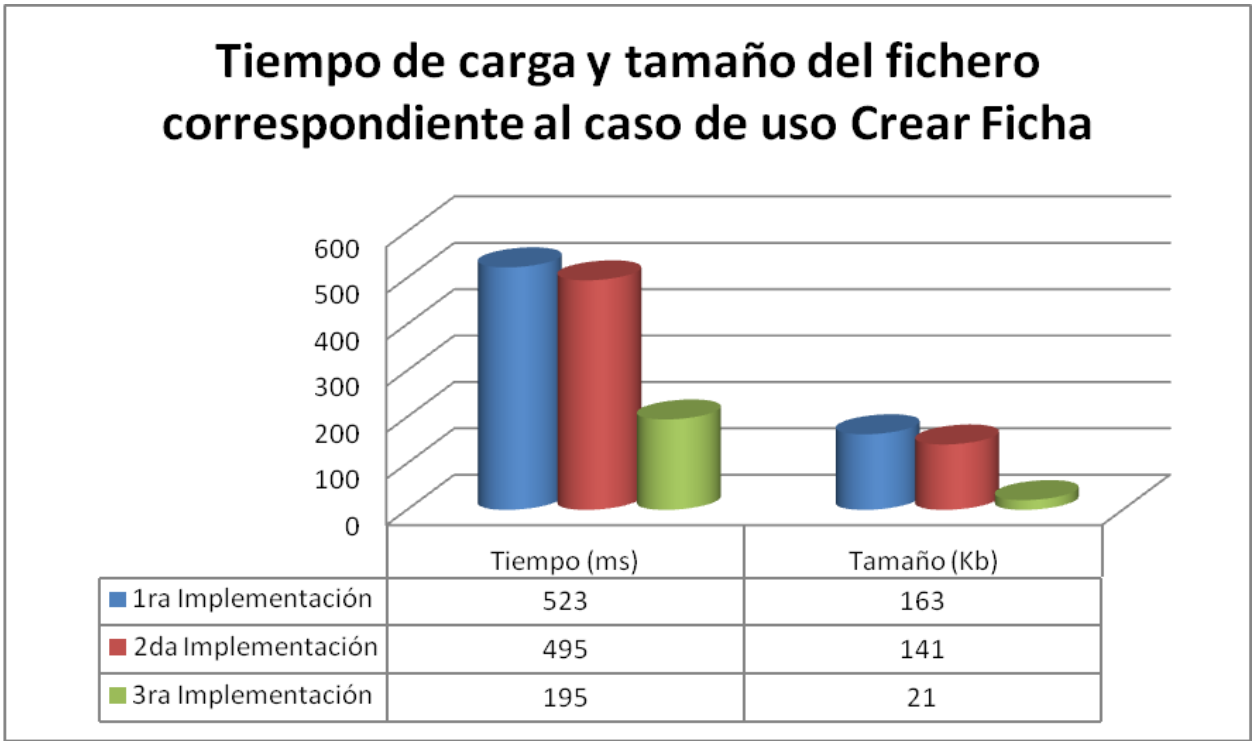


Figura 6 Velocidad de carga y tamaño del fichero correspondiente al caso de uso Crear Ficha.

Los resultados obtenidos en el caso de estudio presentado y en el desarrollo de las interfaces correspondientes a otros casos de uso estudiados, demuestran la factibilidad de la implementación haciendo uso de los componentes visuales presentados en este trabajo, los cuales crecen en funcionalidad y estabilidad proporcionalmente a la creación de nuevas interfaces, que revelan problemáticas a ser incluidas sus soluciones en la implementación de dichos componentes.

## 4.2 Análisis del diseño de interfaces utilizando los componentes visuales

Teniendo en cuenta los parámetros para medir la eficiencia del trabajo con los componentes como parte de la arquitectura de dominio específico del GINA, y realizado un análisis de las interfaces construidas hasta el momento que utilizan estos componentes, se presenta una estimación del tiempo que demora un desarrollador poco capacitado para diseñar vistas que incluyen al menos uno de los componentes, en comparación con las mismas vistas construidas anteriormente sin el uso de estos.

| Casos de Uso | Tiempos de desarrollo (días) | |
|---|---|---|
| | Sin Componentes | Usando Componentes |
| CUCrearFicha | 7 | 2 |
| CUInsertarTrabajador | 7 | 2 |
| CUModificarDatos | 5 | 1 |
| CUResultadosUCESA | 4 | 1 |
| CUEstablecerNecesidadDeConvocatoria | 4 | 1 |
| CUPlanificarPsicometrico | 4 | 1 |
| CUCrearAgenda | 3 | 1 |
| CUPendientesInvestigacion | 3 | 1 |
| CUResultadosPorProcesos | 3 | 1 |
| CUAprobadosPorUnidad | 3 | 1 |
| CUCantidadPorMilitancia | 2 | 1 |
| CUCantidadPorRaza | 2 | 1 |
| CUCantidadPorEscolaridad | 2 | 1 |

Tabla 13 Comparación entre los tiempos estimados para el desarrollo de interfaces de casos de uso.

Los datos presentados en la tabla anterior demuestran la veracidad del rápido diseño de la capa de presentación incorporando el uso de componentes visuales. Además de estimar los tiempos de desarrollo de las interfaces, se pueden obtener otros datos relevantes respecto a la simplificación y reutilización del código de la implementación, que de igual manera ejemplifican la ventaja del trabajo orientado a componentes.

En la siguiente tabla se muestra un conjunto de datos referentes a la cantidad de líneas de código y tiempos de creación de instancias de los componentes presentados, en comparación con requerimientos visuales desarrollados anteriormente sin el uso de estos:

| Requerimientos | Desarrollo Anterior / Componentes | Líneas de Código | Tiempos de desarrollo |
|---|---|---|---|
| Cargar los datos de la tabla TC_PAIS en un ComboBox. | Interfaz Anterior | 21 | 15 min |
| | TcCombo | 4 | 1 min |
| Cargar los datos de la tabla TC_PROVINCIA, y luego de haberse seleccionado una provincia cargar los datos correspondientes de la tabla TC_MUNICIPIO. | Interfaz Anterior | 60 | 1 h |
| | TcRefCombo | 10 | 2 min |
| Cargar, adicionar, modificar y eliminar los datos de contactos de una persona. | Interfaz Anterior | 678 | 10 h |
| | FormGridPanel | 145 | 1 h |
| Obtener los candidatos aprobados en Selección dado un rango de fechas y la unidad perteneciente. | Interfaz Anterior | 217 | 2 h |
| | SearchPanel | 87 | 30 min |
| Guardar el listado de los candidatos aprobados en Selección en un documento con formato digital PDF. | Interfaz Anterior | 40 | 40 min |
| | XExporter | 7 | 1 min |
| Cargar, adicionar, modificar y eliminar la foto de un candidato. | Interfaz Anterior | 35 | 30 min |
| | PhotoPanel | 2 | 1 min |
| Cargar los listados de fichas pendientes y planificadas, | Interfaz Anterior | 395 | 8 h |

| | | | |
|---|---|---|---|
| permitir mover las fichas pendientes al listado de fichas planificadas, y viceversa. | GridToGridPanel | 95 | 1 h |
| Cargar dinámicamente los datos generales de un candidato. | Interfaz Anterior | 160 | 4 h |
| | ColumnFieldset | 50 | 30 min |

Tabla 14 Comparación entre interfaces creadas anteriormente y el diseño de estas utilizando componentes.

Como se puede apreciar en la tabla antes expuesta, el desarrollo de los requerimientos de interfaz basado en los componentes presentados, agiliza considerablemente la implementación de las interfaces estandarizando tanto las funcionalidades como la forma de visualización, y disminuye el tamaño del código propiciando la velocidad de carga de los diferentes módulos para la capa de presentación.

## CONCLUSIONES

En el presente trabajo se abordaron en un principio las problemáticas que originan los objetivos del mismo, argumentadas por el tardío proceso de creación de las interfaces debido a la falta de capacitación de los desarrolladores, y la repetición de código dada por la casi nula reusabilidad de la implementación.

Se identificaron varias características y comportamientos comunes de diferentes requerimientos de la capa de presentación, y se realizó la implementación de once componentes que comprenden dichas características y comportamientos de forma dinámica y reutilizable.

Se presentó la descripción de los componentes visuales implementados, así como de sus atributos de configuración, métodos y eventos.

Se demostró con datos de velocidades de carga, líneas de código, tiempos de desarrollo y tamaños de ficheros, que el desarrollo de la aplicación del GINA basado en la POC disminuye los costos de producción, y abstrae a los desarrolladores de funcionalidades ya implementadas.

## RECOMENDACIONES

- Desarrollar los componentes que no fueron abordados por cuestiones de tiempo.
- Implementar una interfaz de consulta para los desarrolladores que sirva de ayuda para la utilización de los componentes desarrollados.
- Definir un estándar para documentar los componentes sobre la base del modelo Dublin Core Metadata.
- Definir una estructura semántica de representación de los componentes que permita presentar el mecanismo semiautomático para su utilización.

## BIBLIOGRAFÍA


1. **Alvarez Garcia, Mario Alberto. 2009.** *Estándares de soluciones a los problemas de presentación en el Polo de Sistemas Tributarios y de Aduanas.* 2009.
2. **definicionabc. 2009.** http://www.definicionabc.com/tecnologia/sistema-abierto.php. [En línea] 2009. http://www.definicionabc.com/tecnologia/sistema-abierto.php.
3. *Desarrollo de Software Basado en Componentes*
4. **Duffy, Daniel J. 2004.** *Domain Architectures Models and Architectures for UML Applications.* 2004.
5. **encyclopedia2.thefreedictionary. 2010.** http://encyclopedia2.thefreedictionary.com. [En línea] 2010. http://encyclopedia2.thefreedictionary.com.
6. **enunblog. 2008.** http://www.enunblog.com/distribuidos2008/2011/Definici%C3%B3n+de+Sistemas+Distribuidos.html. [En línea] 2008. http://www.enunblog.com/distribuidos2008/2011/Definici%C3%B3n+de+Sistemas+Distribuidos.html.
7. **F. Bertoa, Manuel, Troya, Jose M. y Vallecillo, Antonio.** *Aspectos de Calidad en el Desarrollo de Software basado en componente.*
8. **Falgueras Campderrich, Benet. 2003.** *Ingenieria del software.* 2003.
9. **http://developer.yahoo.com/yui/. 2010.** *http://developer.yahoo.com/yui/.* [En línea] 2010.
10. **http://extjs.com/. 2010.** *http://extjs.com/.* [En línea] 2010.
11. **http://jquery.com/. 2010.** *http://jquery.com/.* [En línea] 2010.
12. **http://www.dojotoolkit.org/. 2010.** *http://www.dojotoolkit.org/.* [En línea] 2010.
13. **http://www.prototypejs.org/. 2010.** *http://www.prototypejs.org/.* [En línea] 2010.
14. **Tracz, Will. 1995.** DSSA (Domain-Specific Software Architecture). [En línea] 1995. http://portal.acm.org/citation.cfm?id=219318.

## ANEXOS


Figura 7 Ejemplo visual del componente PhotoPanel.


Figura 8 Ejemplo de formulario utilizando el componente ColumnFieldset.

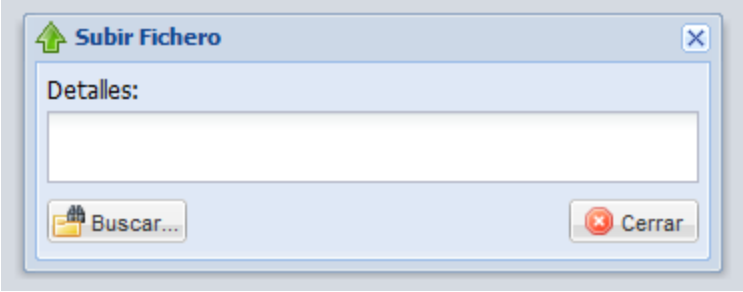


Figura 9 Ejemplo visual del componente UploadFile.

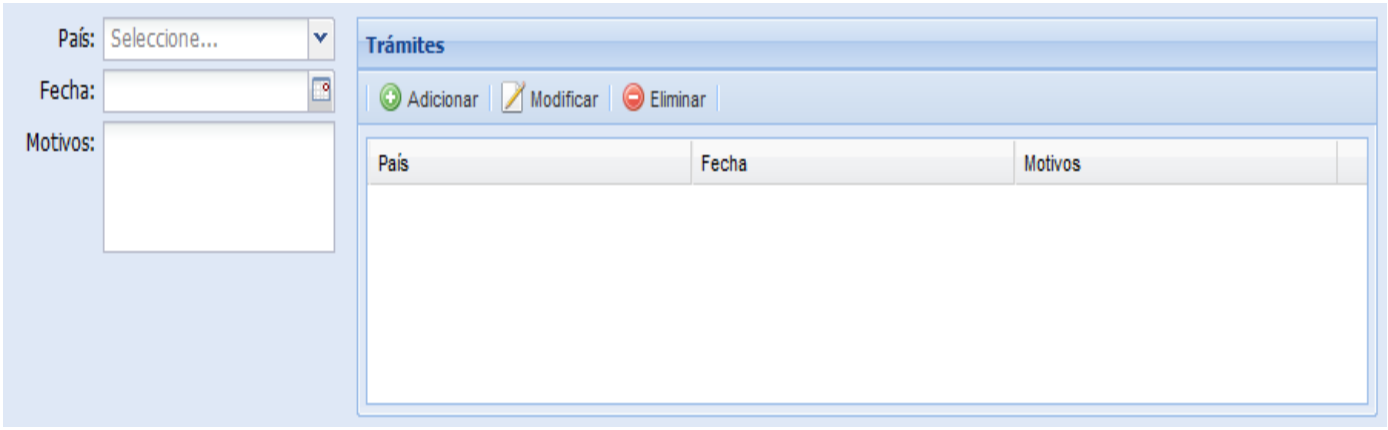


Figura 10 Ejemplo del uso del componente FormGridPanel.

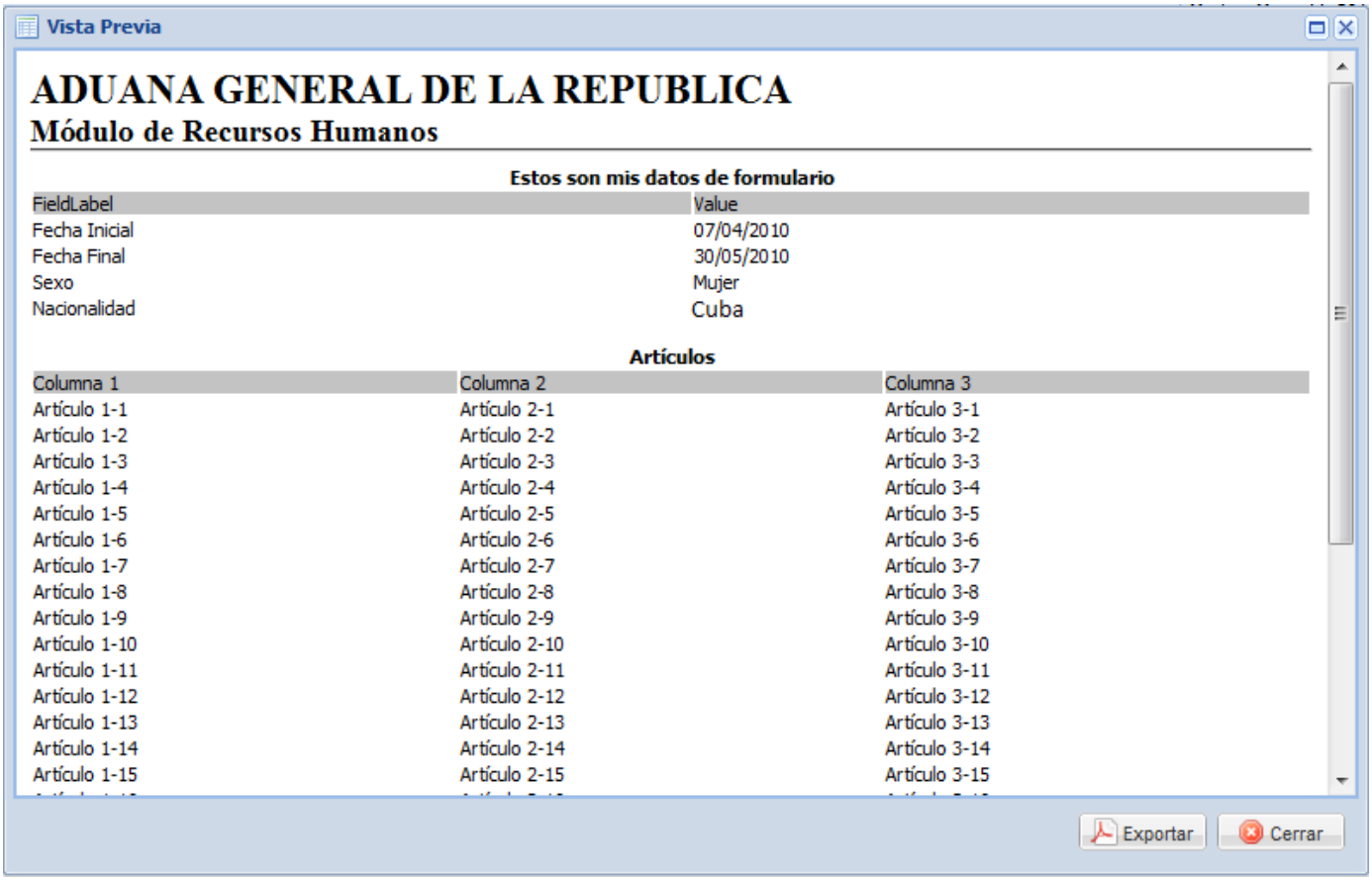


Figura 11 Ejemplo de vista previa que muestra el componente XExporter.

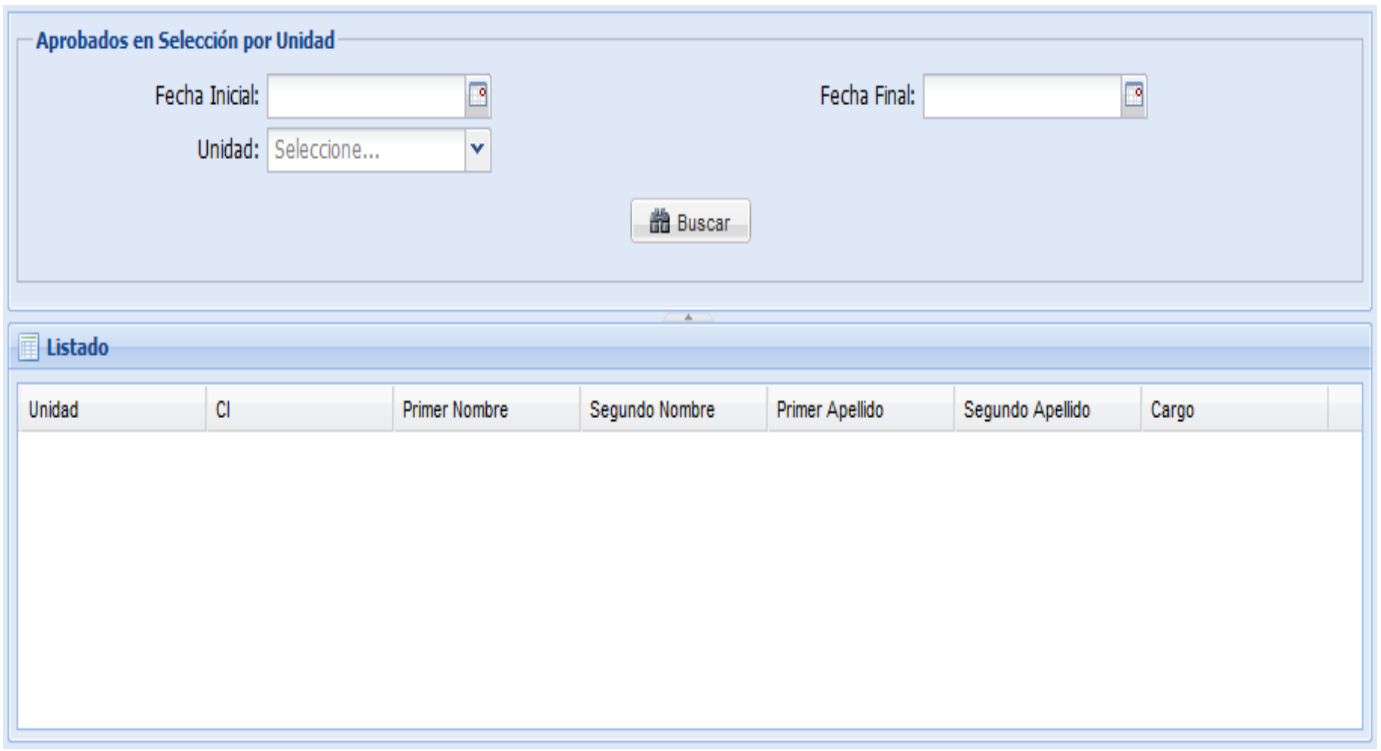


Figura 12 Ejemplo de búsqueda utilizando el componente SearchPanel.

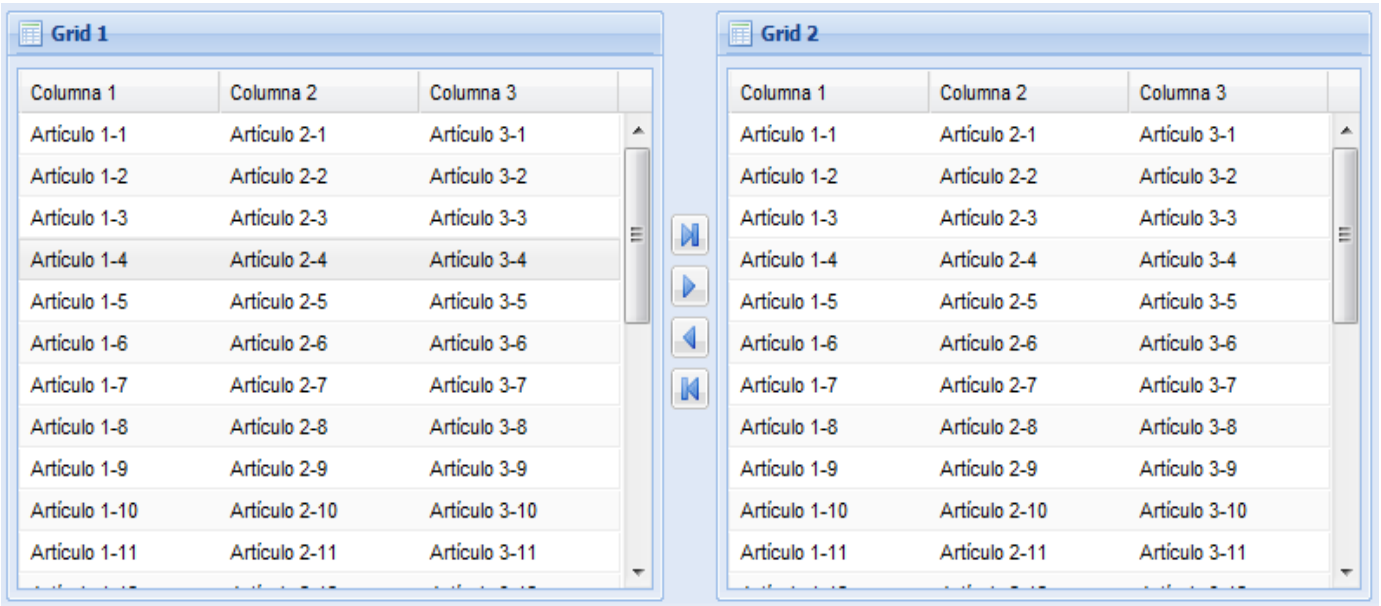

Figura 13 Ejemplo visual del componente GridToGridPanel.

## GLOSARIO DE TÉRMINOS

**Software**: Programas o elementos lógicos que hacen funcionar un ordenador o una red, o que se ejecutan en ellos, en contraposición con los componentes físicos del ordenador o la red.

**Framework:** Es una estructura de soporte definida en la cual otro proyecto de software, puede ser organizado y desarrollado. Típicamente, un framework 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. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

**Javascript:** Es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

**AJAX (Asynchronous Javascript And XML):** Es una técnica de desarrollo web para crear aplicaciones interactivas .Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

**JSON**: (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

**Top-Down**: Técnica de diseño de algoritmos de programación la cual consiste en establecer una serie de niveles de mayor a menor complejidad (arriba-abajo) que den solución al problema. Este diseño consiste en una serie de descomposiciones sucesivas del problema inicial, que recibe el refinamiento progresivo del repertorio de instrucciones que van a formar parte del programa.

**Bottom-Up**: Técnica de diseño de algoritmos de programación, este diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse con forme vayan apareciendo, su análisis como sistema y su codificación, o bien, la adquisición de paquetes de software para satisfacer el problema inmediato.

**XMLHttpRequest:** (XHR), también referida como XMLHTTP (Extensible Markup Language / Hypertext Transfer Protocol), es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores Web. Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas.