UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3

Título: Componente Historial de Personas Naturales para el Sistema GINA.

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

Autor: Yasel Antonio Romero Piñeiro

Tutor: Ing. Julio Cesar Bravo Rodríguez

La Habana, junio de 2011

"Año 53 de la Revolución"

Declaración de Autoría

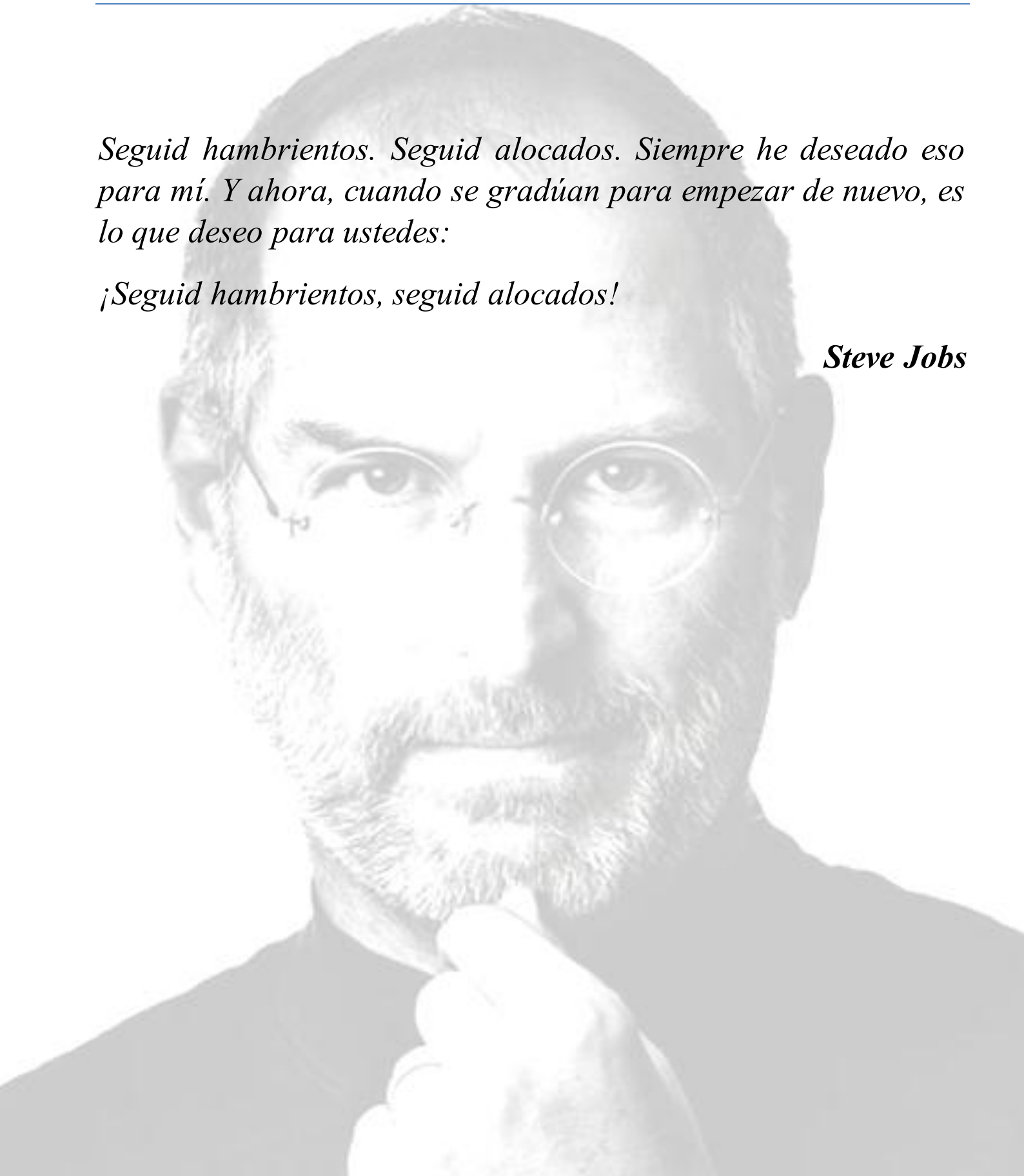
Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Yasel Antonio Romero Piñeiro Ing. Julio Cesar Bravo Rodríguez

Pensamiento



Seguid hambrientos. Seguid alocados. Siempre he deseado eso para mí. Y ahora, cuando se gradúan para empezar de nuevo, es lo que deseo para ustedes:

¡Seguid hambrientos, seguid alocados!

Steve Jobs

Agradecimientos

A la Revolución Cubana por darme la oportunidad de realizar un sueño único.

A mi familia, mi novia y mis amigos por su apoyo y comprensión.

A mi tutor y al equipo de desarrollo del proyecto por dedicarme gran parte de su tiempo.

Dedicatoria

A mis padres, mis hermanos, mis abuelos y mi novia por poder contar con ellos en todo momento.

Al resto de mi familia que estuvo pendiente de mi formación como profesional, brindándome su apoyo y compresión.

A mis amigos por ser mi gran familia en la UCI, por ayudarme incondicionalmente.

Resumen

Tener el control del historial de las personas naturales es una tarea de mucha responsabilidad en las diferentes áreas de trabajo de la Aduana General de la República de Cuba. La presente investigación tuvo como objetivo fundamental concebir un componente que centralice la gestión de este historial, permitiendo una mayor facilidad, accesibilidad y organización a la hora de consultarlo.

Primeramente se realizó un estudio previo de los sistemas que manejan historial de personas naturales en entidades aduanera, analizando sus principales características. Posteriormente se llevó a cabo un estudio de las metodologías y herramientas que fueron usadas para concebir la solución.

Haciendo uso de la Metodología de Desarrollo de Software RUP (en las etapas de diseño, implementación y pruebas) así como de las herramientas estudiadas, se obtuvo un componente que se integró al sistema GINA y que es capaz de centralizar el historial de personas naturales, facilitando de esta forma su consulta en la Aduana General de la República.

Palabras claves: historial, componente, persona natural, GINA.

Tabla de contenidos

Tabla de contenidos

Tabla de contenidos

Introducción

Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación y servir de soporte a la industria cubana de la informática son algunas de las principales misiones a las que está encaminada la Universidad de las Ciencias Informáticas (UCI). Con el surgimiento de esta universidad en el año 2002, Cuba abría las puertas a una nueva escuela que posteriormente se convertiría en una industria de producción de software, brindándole sus servicios a empresas nacionales e internacionales.

La Aduana General de la República (AGR) fue una de los primeros organismos que establecieron relaciones con la UCI para, apoyándose en esta, informatizar los principales procesos que allí se realizan. Precisamente, ese es el principal objetivo de cada uno de los proyectos productivos que mantienen la AGR con la UCI. Como resultado final se pretende obtener un Sistema Integral de Gestión de Aduanas (GINA), manteniendo a la AGR como referencia para las demás aduanas del mundo.

La AGR es una organización creada el 5 de febrero de 1963 con el objetivo fundamental de garantizar la seguridad nacional, es la encargada de regular el control aduanero aplicable a la entrada, el tránsito, el cabotaje, el trasbordo, el depósito y la salida del territorio nacional de mercancías, viajeros y sus equipajes, bienes y valores sujetos a regulaciones especiales, incluidas la flora y la fauna protegidas, así como los medios en que se transporten, además forma parte de la Administración Central del Estado y se subordina al Consejo de Ministros. (1)

Sin una eficiente administración de aduanas, los gobiernos no tendrían claramente definidos sus objetivos frente a la globalización y el continuo crecimiento del comercio internacional, pues la interdependencia económica que vincula y norma las relaciones entre Estados demanda de políticas aduaneras, que en su diseño y aplicación sólo se garantiza su efectividad contando con un personal altamente calificado y con una tecnología apropiada. (2) Por esta razón, la AGR busca elevar continuamente la excelencia de sus servicios, siendo uno de los organismos que más aportan a la economía del país.

En los diferentes flujos de trabajo de la Aduana intervienen personas, ya sean personas jurídicas (o morales) que no son más que aquellas empresas o entidades con las cuales se relaciona la aduana; o personas naturales (o físicas) que se corresponden con las personas de existencia visible, real, física o

natural que intervienen en los distintos procesos. (3) En estas últimas se encontrará centrado el desarrollo de la presente investigación.

Las personas naturales en la AGR se organizan por tipos, formando una estructura que permite un mejor control sobre dichas personas. Estos tipos están dados en dependencia del área a que pertenezcan las personas que en el se agrupan. Entre los tipos más comunes se pueden citar los pasajeros, los tripulantes y los trabajadores. Es importante destacar que una persona puede estar sujeta a más de un tipo.

En busca de una mejor organización de las persona naturales, la aduana efectúa un registro de las principales acciones de dichas personas en su paso por las distintas áreas. A este registro se le conoce como historial de personas.

Como en todas las entidades, la AGR organiza sus plazas de forma jerárquica. Esta estructura permite que exista un elevado número de subordinados, cuyo desempeño laboral puede ser consultado por los jefes en un determinado momento. Conocer el historial de cualquier trabajador en la aduana, si se cuenta con este, puede ser de gran importancia. Esto puede repercutir directamente en la decisión de un jefe sobre sus subordinados así como contribuir al mejoramiento de la calidad del personal con que se cuenta.

Contar con un registro histórico de las incidencias cometidas por las personas en la aduana es de gran importancia para la dirección de esta institución debido a que se puede tener, en cualquier momento, un resumen de los infractores así como de los objetos sobre los cuales se cometieron las incidencias; siendo el historial de cada persona la evidencia principal para llevar el control.

Estar al tanto del comportamiento de los pasajeros que entran y salen de Cuba es otra de las tantas facilidades que se tienen al contar con un historial bien detallado de las personas. A través de ello se puede saber cuántas veces han entrado al país, qué infracciones han cometido en cada una de ellas, además, se pueden determinar posibles sospechosos a partir del análisis de la procedencia o de la nacionalidad de estos.

En fin, contando con un historial bien detallado y fácil de acceder se pueden evitar incidentes que puedan afectar directamente a la economía o comprometer la reputación y el crédito del país.

El proceso de gestión del historial y el registro de las incidencias de las personas naturales en la AGR se realiza de forma manual en estos momentos, es decir, no se cuenta con una herramienta capaz de gestionar los procesos mencionados anteriormente, por lo que la búsqueda del histórico de una

Introducción

persona puede tornarse complicado en un determinado momento que se desee consultar debido al empleo excesivo de tiempo que se requiere.

El sistema GINA se encuentra compuesto por varios subsistemas los cuales se encargan de gestionar los procesos principales que se llevan a cabo en la AGR. Cada subsistema, además de otras tareas, tiene la responsabilidad de gestionar el historial de las personas en su radio de acción, por lo que cada uno de estos historiales se almacenará en un lugar diferente y para consultarlo se deberá hacer desde el propio subsistema. Este proceso se torna algo complicado debido a que la información se encuentra dispersa por los distintos subsistemas, por lo que se desea una solución que centralice el historial de las personas de interés para la AGR facilitando de esta forma el acceso a este por las personas interesadas.

Una vez analizada la situación anterior se formula el siguiente **problema a resolver**: ¿Cómo centralizar el historial de personas naturales en el sistema GINA para facilitar su consulta en la AGR?

Para dar solución al problema planteado anteriormente se tiene como **objeto de estudio:** Los sistemas informáticos para la gestión del historial de personas naturales en entidades aduaneras.

Además, se define como **objetivo general**: Realizar el diseño e implementación de un componente para centralizar el historial de personas naturales en sistema GINA de la AGR; siendo el **campo de acción:** Componente Historial de Personas Naturales en el sistema GINA de la AGR.

Con el desarrollo e implementación del sistema informático Historial de Personas Naturales, se logrará centralizar el historial de personas de interés para la AGR en el sistema GINA.

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

- Estudiar la evolución y el comportamiento de los sistemas informáticos similares al que se desea obtener en la investigación.
- Estudiar las tecnologías y lenguajes existentes para ser utilizadas en el desarrollo de la solución.
- Elaborar el Modelo de Datos del componente Historial de Personas Naturales para el sistema GINA.
- Elaborar el Diagrama de Clases del Diseño del componente Historial de Personas Naturales para el sistema GINA.

- Elaborar los Diagramas de Secuencia de la solución del componente Historial de Personas Naturales para el sistema GINA.
- Elaborar el Diagrama de Componentes para el componente Historial de Personas Naturales para el sistema GINA.
- Obtener el Código de la implementación del modelo escogido para el componente Historial de Personas Naturales para el sistema GINA.
- Realizar la validación de la solución propuesta a través de la realización de pruebas de software.
- Documentar las pruebas realizadas.

Durante el desarrollo de este trabajo son utilizados algunos métodos científicos de la investigación que ayudan a guiar exitosamente el desarrollo de la solución.

- **Histórico-lógico:** con el objetivo de realizar un estudio del estado del arte del tema a investigar. De esta manera se puede conocer acerca de la existencia y características de sistemas de este tipo que hayan sido desarrollados anteriormente.
- **Analítico-sintético:** con el objetivo de analizar teorías y documentos, y a partir de ello realizar un estudio previo acerca de las características y la necesidad de un componente que facilite la búsqueda del historial de personas naturales en la AGR. (4)
- **Modelación:** con el objetivo de realizar abstracciones para dar una explicación de la realidad existente. Su condición principal es la relación entre el modelo y el objeto que se modela, que en este caso sería el sistema en cuestión.
- **Implementación:** con el objetivo fundamental de desarrollar el sistema modelado, implementando las clases que fueron modeladas durante la etapa de diseño. (5)

El presente documento se encuentra compuesto por tres capítulos:

El **capítulo número uno** abarcará todo lo relacionado con la Fundamentación Teórica de la investigación. En este se realizará un análisis del estado del arte a nivel nacional e internacional de los sistemas similares a los que se prevé obtener como resultado final. Además, se tratarán aspectos fundamentales para la comprensión del sistema que se desea modelar, los conceptos más importantes, las tendencias actuales, las técnicas y tecnologías, entre otros aspectos de vital importancia para el desarrollo de la investigación.

De la misma forma, el **capítulo número dos** tratará acerca del Diseño e Implementación del Sistema. Se enfatizará en los diagramas de clases del negocio, el diagrama de paquetes, así como el diagrama

de entidad relación de la base de datos incluyendo la descripción de las tablas, además de los diagramas de componente y secuencia de la solución.

Finalmente, en el **capítulo número tres** se hará la Validación de la Solución propuesta, haciendo uso de las distintas técnicas de evaluación de software, para determinar posibles no conformidades así como saber si se cumplió o no con los objetivos propuestos.

Capítulo 1. Fundamentación Teórica

1.1 Introducción

En el presente capítulo se describen los elementos principales que fundamentan el contenido de la investigación. Se realiza una descripción de varios sistemas que se utilizan para gestionar el historial de las personas naturales en entidades aduaneras, argumentando sus ventajas y desventajas. Se detallan aspectos importantes de las distintas tecnologías que serán empleadas durante el desarrollo del sistema.

1.2 Las aduanas

El Real Academia Española define a las aduanas como una oficina pública, establecida generalmente en las costas y fronteras, para registrar, en el tráfico internacional, los géneros y mercaderías que se importan o exportan, y cobrar los derechos que adeudan. (6)

Actualmente es posible afirmar que las aduanas constituyen el principal organismo ejecutor de la política de comercio internacional del Estado en cuanto al control y cumplimiento de las regulaciones económicas, administrativas, contractuales, restrictivas y tributarias que afectan los términos físicos del intercambio. (7)

1.2.1 La aduana en Cuba

En Cuba, la aduana constituye un órgano de control en la frontera y de fiscalización en la actividad vinculada al comercio exterior. Entre sus misiones está garantizar la seguridad y protección de la sociedad socialista y de la economía nacional, así como la recaudación fiscal y la emisión de las estadísticas del comercio exterior, a través del cumplimiento de las políticas estatales de competencia aduanera para el tráfico internacional de viajeros, mercancías y medios de transporte. (1)

1.3 Historial de Personas Naturales

La Real Academia Española define que el historial no es más que una reseña circunstanciada de los antecedentes de algo o de alguien. (8)

En algunos lugares, generalmente centros de trabajo, se asocia al historial de las personas naturales con su currículum vítae[1] resumido en una página que generalmente acompaña a una solicitud de empleo.

[1] **Currículum vítae**: relación de los títulos, honores, cargos, trabajos realizados, datos biográficos, etc., que califican a una persona.

En la AGR el historial de las personas naturales es un resumen que incluye las principales acciones (incidencias) que han cometido dichas personas en la aduana; estas personas pueden estar asociadas a pasajeros o trabajadores.

Este historial es un aspecto que se debe tener en cuenta cuando se desee valorar los indicadores correspondientes al comportamiento de dichas personas. En la actualidad, las técnicas de recopilación de los antecedentes de las personas naturales se han sofisticado de manera que se usan, en mucho de los casos, programas informáticos para facilitar dicho proceso. Algunos de los principales Sistemas Integrales de Gestión de Aduanas son muestra de ello.

1.3.1 Sistemas informáticos en el mundo

A continuación se muestran algunos de los principales Sistemas Integrales de Gestión de Aduanas que contienen herramientas para gestionar el historial de las personas naturales.

SIDUNEA

El Sistema Aduanero Automatizado (SIDUNEA) es la herramienta informática para el control y administración de la gestión aduanera, desarrollada por la Conferencia de las Naciones Unidas sobre el Comercio y el Desarrollo (UNCTAD), y que actualmente es usada con éxito en más de 80 países.

Permite realizar un seguimiento automatizado de las operaciones aduaneras y controlar efectivamente la recaudación de los impuestos aduaneros, porque este sistema verifica automáticamente los registros, calcula los impuestos y contabiliza todo lo relativo a cada declaración, con la mínima intervención del factor humano subjetivo. Al ser un sistema multidisciplinario, está especializado en cada área del trabajo aduanero para ser la herramienta de trabajo de todos los clientes de la aduana, sean usuarios internos o externos, privados o públicos.

Entre sus funciones principales se encuentran la Auditoría y Gestión del Historial de las personas que intervienen en los distintos procesos aduaneros. (9)

A pesar de ser una de las aplicaciones que mejor maneja los procesos aduaneros en el mundo presenta una serie de limitaciones para ser desplegado en Cuba. Al igual que en muchos países del mundo, en Cuba se llevan a cabo procesos aduaneros propios, lo que limita un funcionamiento satisfactorio de cualquier software que se desee aplicar debido a que tiene que adaptarse a características y condiciones de funcionamiento totalmente desconocidas para él. Otra de las limitaciones que posee este software es su carácter privativo, con licencias que se acercan a los ocho millones de dólares al año, factor que imposibilita el acceso de los países subdesarrollados.

Korea Custom Service (KCS)

El Sistema de Servicios Aduanales de la República de Korea trabaja sobre la base de los sistemas EDI (Intercambio Electrónico de Datos), logrando la transferencia por medios electrónicos de información comercial o de negocios. Presenta un eficaz Servicio de Aduanas mediante el establecimiento de un Sistema de Liquidación preciso y rápido. Mejora la competitividad nacional al reducir el tiempo y costo requerido para el despacho de aduana, y la reducción de los costos logísticos. Mejora también los servicios públicos, ofreciendo la información relativa a la Administración de Aduanas de manera precisa y rápida.

Esta solución presenta una serie de características que lo sitúan entre los primeros de su tipo en el mundo. Desarrolla un Sistema de Información, Investigación y Vigilancia para el enfrentamiento a acciones delictivas que se cometan en las aduanas. Este sistema de información se compone de cinco subsistemas: Infamación General, Actividades de Investigación sobre Infracciones, Área de Investigación de la Empresa, Área de Vigilancia de Pasajeros, Vigilancia de los Buques y la Tripulación. Todas estas áreas en conjunto se encargan principalmente de la gestión de ex-convictos y sospechosos de violaciones en las aduanas, de la notificación a las autoridades en caso violaciones así como la gestión de la información sobre los pasajeros de alto riesgo y de viajeros frecuentes. (10)

El KCS cuenta con una serie de limitaciones si se analiza desde el punto de vista de un futuro despliegue en Cuba. Su carácter privativo va en contra de la tendencia cubana a la migración del software libre. Otra de las limitaciones es que el registro de las incidencias se basa sólo en los pasajeros y no toma en cuenta las incidencias cometidas por los trabajadores de la aduana. Además, aunque es un sistema bastante adaptable se hace muy difícil adaptarlo a los procesos aduaneros propios de Cuba.

DDS

DDS es una empresa perteneciente a la República de Argentina cuyo propósito principal es el desarrollo de software para comercio exterior; sistemas de gestión para despachantes de aduana, importadores, exportadores, agentes de carga internacional y gestión de depósitos en zona franca. (11)

El software de gestión de aduanas desarrollado por esta empresa es un sistema que se encarga principalmente del control del comercio exterior. Desarrolla un Sistema de Gestión para Exportadores

el cual incluye la trazabilidad de sus despachos temporales, descargando de manera automática o manual los insumos cuando realiza el registro de los permisos de embarque.

Este sistema implementa un estricto control de las incidencias llevadas a cabo durante los distintos procesos que rigen el comercio exterior. Su principal desventaja consiste en que a pesar de incursionar otras áreas de la gestión aduanera se centra principalmente en la gestión del comercio exterior dentro de las aduanas, además es un software privativo y los costos de las licencias están en dependencia de los acuerdos tomados entre ambas partes. Las desventajas mencionadas anteriormente hacen que este sistema no sea idóneo para llevar a cabo la gestión de los procesos en la AGR.

Actualmente en la AGR se no cuenta con un sistema capaz de centralizar la gestión de las incidencias cometidas en los diferentes procesos que se llevan a cabo, de ahí la necesidad de desarrollar una solución que permita gestionar el historial de las personas naturales en la aduana y que se adapte a las características de Cuba.

1.4 Descripción de la solución

El sistema GINA es la solución de software cubana para la gestión de los procesos aduaneros. Es desarrollado por la UCI en trabajo unido a los especialistas del Centro de Automatización de la Dirección y la Información (CADI). Este trabajo en conjunto se inició en el año 2004 y dentro de los planes se encuentra el desarrollo de un Componente Persona que se encarga de la gestión las propiedades fundamentales de las personas en el sistema GINA. Funciona como un eslabón común entre el resto de los subsistemas, brindando servicios a cada uno de ellos. Precisamente, uno de los módulos de dicho componente es el encargado de gestionar el historial de las personas naturales dentro del sistema GINA.

Dicho módulo servirá para centralizar el historial de cada uno de los subsistemas del GINA haciendo que este pueda consultarse más rápidamente y de forma organizada permitiendo tener un resumen de las incidencias cometidas por las personas en los distintos procesos que se llevan a cabo en la aduana.

1.5 Tendencias actuales y tecnologías empleadas.

En la actualidad el uso de aplicaciones web[2] representa un elemento que ha contribuido considerablemente al desarrollo de numerosas empresas. Este tipo de aplicaciones presenta ventajas

[2] **Aplicación web**: son aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador web.

significantes en cuanto al ahorro de tiempo de los procesos, compatibilidad y disponibilidad de la información.

Durante estudios previos al desarrollo del sistema GINA se definió la línea base de la arquitectura la cual se compone por las principales tecnologías empleadas en el desarrollo del sistema. A continuación se muestra una descripción de algunas de dichas tecnologías haciendo énfasis en sus principales características.

1.5.1 Metodología de desarrollo

Como metodología de desarrollo se utiliza el Proceso Racional Unificado (RUP). Esta metodología propone utilizar el Lenguaje Unificado de Modelado (UML), donde forman la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. La misma constituye una guía rectora que decide quién hace qué, cuándo y cómo lo hace. RUP dirige las actividades que se realizan por roles, les da un orden, define qué artefactos deberían ser desarrollados y puede además monitorear y medir los productos y actividades de un proyecto, teniendo en cuenta la calidad del producto final. RUP se caracteriza por dividir el ciclo de vida de la producción del software en 4 fases:

1. **Inicio o Conceptualización:** es donde se determina la visión del proyecto, o sea se comprende el entorno y se determina el alcance del producto.
2. **Elaboración:** en esta etapa se determinan los cimientos de la arquitectura y se analiza el dominio del problema.
3. **Construcción:** en esta fase se obtiene la capacidad operacional inicial del producto.
4. **Transición:** Se obtiene liberación del producto y se pone en manos de los usuarios finales. (12)

Además de esto cuenta con 9 flujos de trabajo, 6 de ingeniería y 3 de soporte los cuales son: Modelamiento del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Despliegue, Gestión de configuración y cambios, Gestión de proyectos y por último Entorno, donde los 3 últimos constituyen los flujos de soporte. Esta metodología puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicaciones, diferentes tipos de organizaciones, diferentes niveles de amplitud y diferentes tamaños de proyectos.

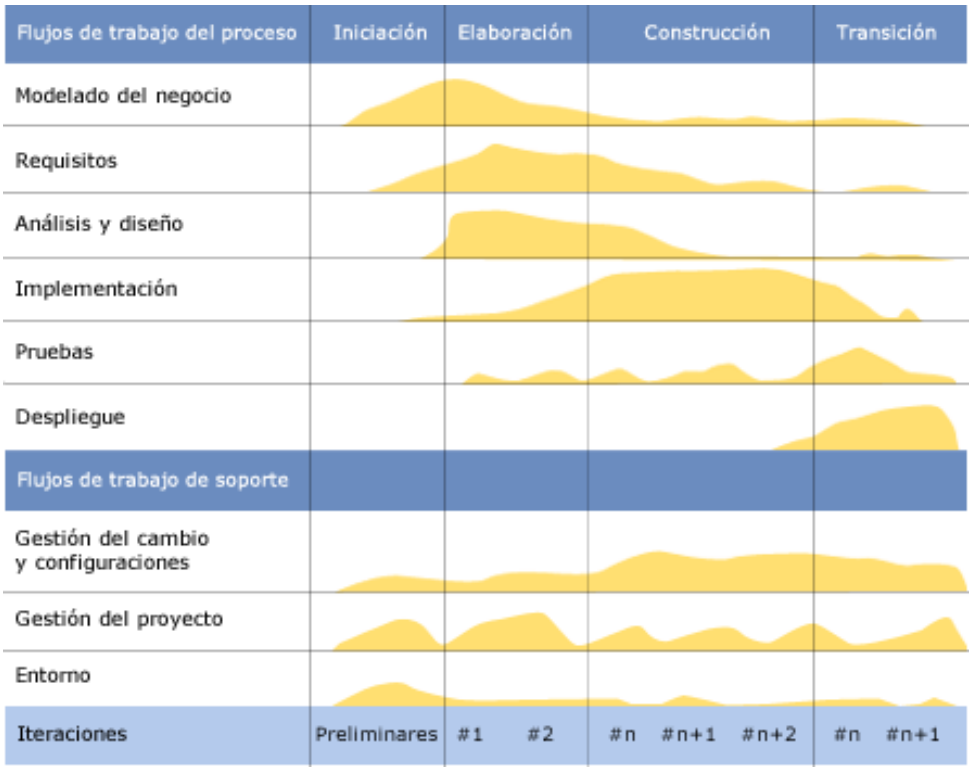


Figura 1. Fases y Flujos de Trabajo de RUP.

Los elementos característicos de RUP son:

- **Actividades**: son los procesos que se llegan a realizar en cada iteración.
- **Trabajadores**: son las personas o entidades involucradas en cada proceso.
- **Artefactos**: un artefacto puede ser un modelo, o un elemento de modelo, un documento, en fin todo lo que puede ser generado en el proceso.
- **Flujo de Actividades**: secuencia de actividades realizadas por trabajadores que producen un resultado de valor observable.

Teniendo en cuenta la flexibilidad de RUP, en el Departamento de Soluciones para la Aduana se aplica esta metodología con adaptaciones especiales. El impacto de dichas adaptaciones afectará el desarrollo de la presente investigación durante la etapa de diseño, permitiendo la elaboración de diagramas de secuencias orientados a actividades[3].

[3] Se profundizará sobre el uso de este tipo de diagrama durante el desarrollo del capítulo dos.

1.5.2 Metodologías de Diseño

Metodología de Diseño se refiere al desarrollo de un sistema o método para una situación única. Hoy en día, el término suele aplicarse a los campos tecnológicos en referencia al diseño web, software o diseño de sistemas de información. A continuación de describen algunas de estas tecnologías que son usadas en el desarrollo de la solución.

Diseño Orientado al Uso: Cuando el diseño se centra en las tareas asociadas al uso y sus objetivos. Este enfoque penaliza la usabilidad, ya que incrementa la curva de aprendizaje, pero logra solventar problemas complejos o de alta criticidad.

Diseño Centrado en el Usuario: (UCD *User Center Design*) es la metodología de diseño más habitual en el mundo. Este enfoque coloca todas las necesidades, deseos y limitaciones del usuario como núcleo del proceso de diseño. Por lo cual esta metodología conlleva por definición investigación y análisis del usuario. Así mismo implica que el proyecto debe ser distribuido por faces para garantizar los resultados. Dentro del UCD hay metodologías específicas, como son KES (*Kansei Engineering System*) o QFD (*Quality Function Deployment*). (13)

Enfoque Ascendente: se refiere a la identificación de aquellos procesos que necesitan computarizarse conforme vayan apareciendo, su análisis como sistemas y su codificación; o bien, la adquisición de paquetes de software para satisfacer el problema inmediato.

Enfoque Descendente: Consiste en representar una imagen del sistema y luego dividirla en fracciones más pequeñas. El diseño descendente es compatible con la manera de pensar sobre los sistemas en general.

Desarrollo Modular:

Su funcionamiento se basa en una descomposición de la programación en fracciones lógicas y manejables. Este tipo de programación se apega bien al diseño descendente porque enfatiza las interfaces entre los módulos, más que mantenerlas ignoradas hasta el final del desarrollo del sistema. Cada módulo debe ser funcionalmente cohesivo, de manera que satisfaga sólo una función. (13)

1.5.3 Patrones de diseño

Un patrón de diseño es una solución estándar para un problema común de programación, una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios, una estructura de implementación que logra una finalidad determinada, manera práctica de describir aspectos de la organización de un programa. (14)

Los patrones brindan la facilidad de reutilizar el conocimiento de desarrolladores, clasificando y describiendo problemas y soluciones a problemas que surgen con frecuencia durante el desarrollo, por lo que se convierten en un historial que ayuda al equipo de desarrollo a no cometer errores ya descritos. (14)

A continuación se muestran los principales patrones de diseño que se emplean en el desarrollo de la solución. Constituyen patrones GRASP[4] y son implementados por el *framework* de desarrollo, elemento que se detallará más adelante.

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

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

1. B contiene A.
2. B es una agregación (o composición) de A.
3. B almacena A.
4. B tiene los datos de inicialización de A (datos que requiere su constructor).
5. B usa A.

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

Bajo Acoplamiento: el acoplamiento es una medida de fuerza con que un elemento está en, tiene conocimiento de, confía en, otros elementos. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. (14)

Alta Cohesión: la cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo. (14)

[4]**GRASP:** describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Controlador: es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos. Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo. (14)

Decorador: se corresponde con un patrón estructural perteneciente a la familia de los patrones GoF[5]. Responde a la necesidad de añadir dinámicamente funcionalidad a un objeto. Esto permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera. (15)

Su aplicación se basa principalmente en los siguientes casos:

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

Figura 2. Implementación del patrón Decorador por el *framework* Symfony.

1.5.4 Patrones Arquitectónicos

Son los que definen la estructura de un sistema de software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema.

Modelo Vista Controlador (MVC), es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El *framework*

[5] **GoF:** *Gang of Four* es el nombre con el que se conoce comúnmente a los autores del libro *Design Patterns*, referencia en el campo del diseño orientado a objetos.

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC. Este patrón divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (16)

En la siguiente figura se muestra la implementación del patrón MVC en Symfony, un *framework* de desarrollo que se estará explicando más adelante.

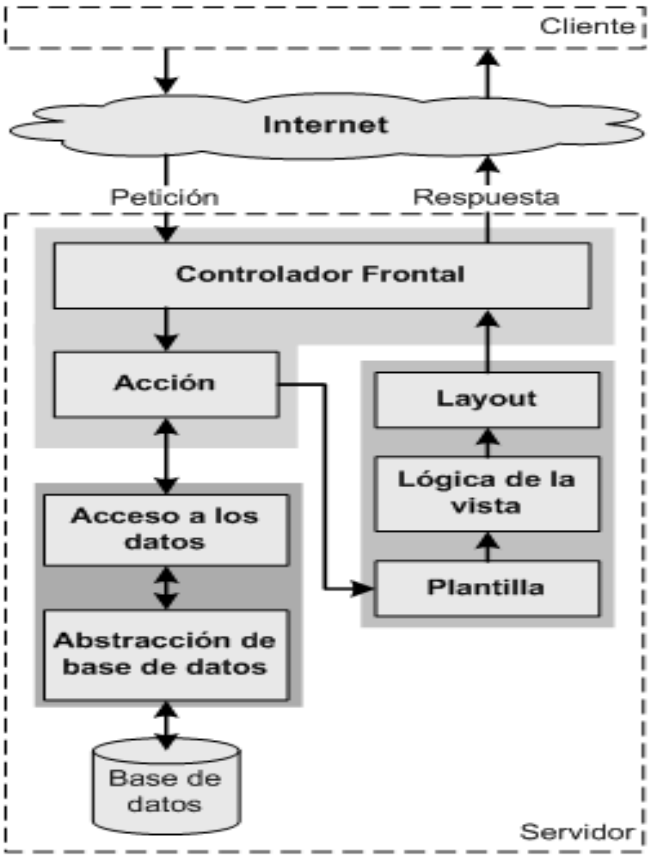


Figura 3. Implementación del patrón MVC por el *framework* Symfony.

1.5.5 Lenguajes de programación

Un lenguaje de programación no es más que un conjunto de sintaxis y reglas semánticas que definen los programas del computador. Es una técnica estándar de comunicación para entregarle instrucciones al computador. Un lenguaje le da la capacidad al programador de especificarle al computador, qué tipo

de datos actúan y que acciones tomar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano.

Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, interpretación o intermedio, es decir, ser traducido al lenguaje de máquina para que pueda ser ejecutado por el ordenador. (17)

En la actualidad son muy variados los lenguajes de programación que se usan para desarrollar las distintas soluciones, partiendo desde aquellos privativos por los cuales es necesario pagar una patente y llegando hasta los lenguajes de carácter libre cuyo soporte es dado por la comunidad de usuarios que deseen colaborar.

1.5.5.1 Lenguajes utilizados en el lado del cliente

Para realizar la programación en el lado del cliente[6] fueron elegidos los siguientes lenguajes:

XHTML:

El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado W3C (*World Wide Web Consortium*). Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de la misma manera en cualquier navegador de cualquier sistema operativo. (18)

El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, HTML es descendiente directo del lenguaje SGML[7] (*Standard Generalized Markup Language*), mientras que XHTML lo es del XML (que a su vez, también es descendiente de SGML). (18)

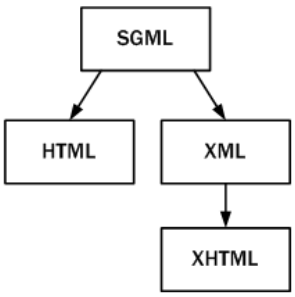


Figura 4. Esquema de la evolución de HTML y XHTML.

[6] **Lenguajes en el lado del cliente:** son los lenguajes que basan su procesamiento en el cliente web, es decir que se ejecutan en el navegador del usuario.
[7] **SGML**: consiste en un sistema para la organización y etiquetado de documentos

En el sistema GINA es utilizada la versión 1.0 del estándar XHTML. Este es el lenguaje básico sobre el cual se encuentra desarrollada la interfaz de usuario del sistema.

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

La separación de los contenidos y su presentación posee numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (19)

En el sistema GINA el lenguaje CSS es empleado para decorar el aspecto de la capa de presentación.

JavaScript:
Constituye un lenguaje de programación del lado del cliente inventado por Brendan Eich en la empresa *Netscape Communications*. Usado en las páginas web para agregar mayor funcionalidad, interacción o animaciones. (20)

Entre sus principales características es posible citar:

- Es interpretado por el cliente.
- Está basado en objetos.
- Su código se integra en las páginas XHTML, incluido en las propias páginas.
- Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto no se compila.
- Es independiente de la plataforma, por lo que se ejecuta en cualquier sistema operativo. (21)

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

1.5.5.2 Lenguajes utilizados en el lado del servidor

El sistema GINA se ha implementado sobre la base del PHP como lenguaje de programación del lado del servidor[8].

[8] **Lenguajes en el lado del servidor:** son los lenguajes que se procesan en un servidor remoto y que generan la página web antes de enviarla al cliente.

El PHP es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. (22)

Cuenta con cuatro grandes características que hacen que este lenguaje sea distinguido entre los demás existentes en el mundo:

- **Velocidad:** no solo la velocidad de ejecución, la cual es importante, sino además no crear demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema. PHP se integra muy bien junto a otro software, especialmente bajo ambientes Unix, generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz.
- **Estabilidad:** ninguna aplicación es 100% libre de errores, pero PHP goza de la ayuda de una gran comunidad de desarrollo, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- **Seguridad:** PHP provee diferentes niveles de seguridad, estos pueden ser configurados de archivos de configuración que se encuentran en el servidor.
- **Simplicidad:** es un lenguaje de programación simple de implementar por lo que usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente. (23)

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

Para el desarrollo del sistema GINA se emplea el lenguaje de programación PHP en su versión 5.2.

1.5.6 Frameworks de trabajo

Se han realizado importantes progresos orientados a la reusabilidad del software a través de la aplicación del paradigma de la programación orientada a objetos y mediante el uso de los componentes tecnológicos. Sin embargo, estas tecnologías sólo proveen el re-uso en el nivel individual, frecuentemente a menor escala. Con la aparición de los patrones de diseño se ha demostrado la reutilización de soluciones para resolver problemas de escala mayor enfocados en la solución de problemas sencillos. Sin embargo, el más complejo problema de la reutilización de soluciones es en el nivel de los grandes componentes, para que estos se puedan adaptar para las solicitudes individuales. (24)

Un *framework*, en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (25)

Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un *framework* proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (16)

1.5.6.1 Framework de trabajo Ext JS

Ext JS es una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de navegadores que nos permite crear páginas e interfaces web dinámicas. (26) Permite realizar aplicaciones web enriquecidas basándose en tecnología AJAX, JSON, DHTML y DOM.

Con Ext JS, se pueden desarrollar aplicaciones web multiplataforma con facilidad. El modelo de componente de Ext JS mantiene su código bien estructurado por lo que incluso las aplicaciones más grandes pueden ser de fácil mantenimiento.

Ext JS brinda la posibilidad de utilizar un gran número de componentes visuales que mejoran considerablemente la calidad de las aplicaciones. Brinda la posibilidad de validaciones de formularios de todo tipo, basándose en expresiones regulares y tipos de datos. Trae implícitos componentes como vista en árboles, arrastrado y soltado, cambio de tamaño de imágenes, rejillas, paginado, agrupado de objetos, asistentes, entre otros.

La utilización de un *framework* de JavaScript como Ext JS facilita la separación de las capas de la vista con la del controlador desde el punto de vista productivo ya que el código utilizado en la primera es solamente JavaScript y no es necesario utilizar ningún tipo de código PHP, así los desarrolladores pueden centrarse más en el aprendizaje de un solo lenguaje. Además, al soportar la serialización de objetos mediante tecnología JSON, permite que los datos enviados desde el controlador como respuesta a la vista contengan sólo las propiedades de dichos objetos, pero no el comportamiento, minimizando los posibles errores de programación y los accidentes de que los objetos sean modificados erróneamente desde la vista. (26)

Para en desarrollo del sistema GINA este *framework* es utilizado en su versión 3.0, principalmente, para validar los datos antes de ser enviados al servidor y para proveer al sistema de una buena apariencia y usabilidad de forma tal que se simule una aplicación de escritorio en el navegador.

1.5.6.2 Framework de trabajo Symfony

Symfony es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja.

Las clases de la capa del modelo se generan automáticamente, en función de la estructura de datos de la aplicación. La librería Propel[9] se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario. (16)

La abstracción de la base de datos es completamente transparente para el programador, ya que se realiza de forma nativa mediante PDO (*PHP Data Objects*). Así, si se cambia el sistema gestor de bases de datos en cualquier momento, no se debe reescribir el código, ya que tan solo es necesario modificar un parámetro en un archivo de configuración.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Su carácter libre y multiplataforma lo hacen uno de los *framework* de PHP más usados en el mundo. (16)

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server)
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

[9] **Propel:** es un proyecto de software libre, es una de las mejores capas de abstracción de objetos/relacional disponibles en PHP 5.

- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (16)

Este *framework* de desarrollo, en su versión 1.2.8, es la herramienta base utilizada para llevar a cabo la implementación del sistema GINA, proporcionando robustez y seguridad al futuro producto de software.

1.5.7 Entornos de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado (en inglés *Integrated Development Environment* o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. (27)

Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.

NetBeans IDE

NetBeans IDE es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE, además es un producto libre y gratuito sin restricciones de uso. (28) Posee un amplio soporte para el lenguaje PHP así como para los *framework* de trabajo Symfony y Ext JS.

Para el desarrollo de la solución se ha elegido NetBeans IDE, en su versión 6.9.1, como Entorno de Desarrollo Integrado.

1.5.8 Herramientas de Diseño

Hoy en día, muchas empresas se han extendido a la adquisición de herramientas CASE[10], con el fin de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema, desde el principio, hasta el final y así incrementar su posición en el mercado competitivo.

[10] **CASE:** siglas en ingles que se utilizan para referirse a Ingeniería de Software Asistida por Computadora

Algunas de estas herramientas tienen un valor económico muy alto y requieren costos de entrenamiento de personal muy altos, además se enfrenta la falta de adaptación de la herramienta, a la arquitectura de la información en la que está compuesta y a las metodologías de desarrollo utilizadas por la organización.

Visual Paradigm for UML

Visual Paradigm for UML (VP-UML) es un modelador UML que permite el diseño de sistemas con todo tipo de tipos de diagramas UML. También es utilizado para diseñar diagramas de casos de uso, diagramas de requerimiento y diseño de bases de datos relacionales. Con VP-UML, el equipo de desarrollo de software puede realizar el análisis y diseño de sistemas con eficacia. (29)

Es una herramienta multiplataforma y se integra con algunas herramientas realizadas en Java entre las cuales se puede citar el NetBeans IDE.

Debido a las características y facilidades argumentadas anteriormente se elige la herramienta Visual Paradigm for UML en su versión 3.4 para la realización del diseño de la solución.

1.5.9 Gestor de Base de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. (30)

En las soluciones implementadas para la AGR el gestor de base de datos que se utiliza es Oracle en su versión 11g. Consiste en un sistema de gestión de base de datos relacional desarrollado por Oracle Corporation. Se considera a este producto como uno de los sistemas de bases de datos más completo en el mercado internacional, destacando el soporte de transacciones, la estabilidad, escalabilidad y soporte multiplataforma.

Principales ventajas de Oracle

- Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia.
- Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente.

- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos.
- Tiene buen rendimiento y hace buen uso de los recursos.
- Posee un rico diccionario de datos.
- Brinda soporte a la mayoría de los lenguajes de programación.
- Es un sistema multiplataforma, disponible en Windows, Linux y Unix.
- Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal. Las copias de la base de datos productiva pueden estar en modo de lectura solamente.

Desventajas

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

La AGR usa este software desde el año 1997, periodo en el cual ha obtenido experiencias positivas en cuanto al uso de dicho producto.

1.6 Conclusiones parciales

El estudio de los principales Sistemas Integrales de Gestión de Aduanas que gestionan el historial de personas naturales en entidades aduaneras permitió sentar las bases que fundamentaron las tecnologías y herramientas a utilizar.

Como línea base de la arquitectura se empleó RUP como metodología de desarrollo de software, Ext JS en su versión 3.0 como *framework* de trabajo en el lado del cliente, Symfony en su versión 1.2.8 como *framework* de trabajo en el lado del servidor y como SGBD se empleó Oracle en su versión 11g.

Capítulo 2. Diseño e Implementación del Sistema

2.1 Introducción

En este capítulo se desarrolla el diseño y la implementación de la solución. El epígrafe de diseño tiene como objetivo fundamental obtener los distintos diagramas que guiarán el proceso de implementación del sistema, profundizando en el modelo de diseño de la solución. El epígrafe de implementación será el encargado de dar una visión de la implementación del sistema, profundizando en el código obtenido para facilitar el entendimiento de este, además se obtendrá el diagrama de paquetes de la solución para facilitar un futuro mantenimiento del sistema así como los estándares de codificación empleados durante la implementación.

2.2 Diseño del Sistema

El diseño constituye el tercer flujo de trabajo de RUP. En este se modela el sistema y se encuentra su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales y otras restricciones, que se le suponen. Los propósitos fundamentales del diseño se describen a continuación.

- Adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, tecnologías de interfaz de usuario, tecnologías de gestión de transacciones, etc.
- Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguientes capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo, teniendo en cuenta la posible concurrencia.
- Visualizar y reflexionar sobre el diseño utilizando una notación común. (31)

Como se explicaba en el epígrafe **1.5.1**, la metodología de desarrollo RUP es utilizada en el desarrollo de la solución con adaptaciones estructurales algunas las cuales se verán durante la etapa de diseño de la presente investigación.

2.2.1 Modelo del Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de usos centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen un impacto en el sistema a considerar. Además,

el modelo de diseño sirve como abstracción de la implementación del sistema y es, de ese modo, utilizado como una entrada fundamental de las actividades de implementación. (31)

El modelo del diseño de la presente investigación se encuentra compuesto por el diagrama de clases del negocio, el diagrama de paquetes y por los diagramas de secuencia orientados a las actividades de los requisitos funcionales.

2.2.1.1 Diagrama de Clases del Negocio

Los diagramas de clases muestran la estructura estática del modelo, en particular, las cosas que existen (como clases y tipos), su estructura interna y sus relaciones con otras clases. Los diagramas de clases no muestran información temporal, aunque puede contener ocurrencias de las cosas que tiene o las cosas que describen el comportamiento temporal. (32)

En la figura 5 se muestra el diagrama de clases del negocio del componente Historial de Personas Naturales para el sistema GINA. Como se puede apreciar, se compone por clases distribuidas en cuatro colores los cuales ayudan a comprender mejor la distribución de estas en el negocio.

Las clases en color gris representan a las entidades nomencladoras y que las mismas se encuentran fuera del esquema correspondiente al historial de personas naturales. Las clases representadas con color amarillo muestran las entidades nomencladoras y que forman parte del esquema historial de personas naturales. La clase de color verde representa la entidad encargada del manejo del registro de las incidencias de personas naturales. Finalmente se encuentra la clase coloreada en blanco encargada del control de las personas naturales dentro del sistema GINA.

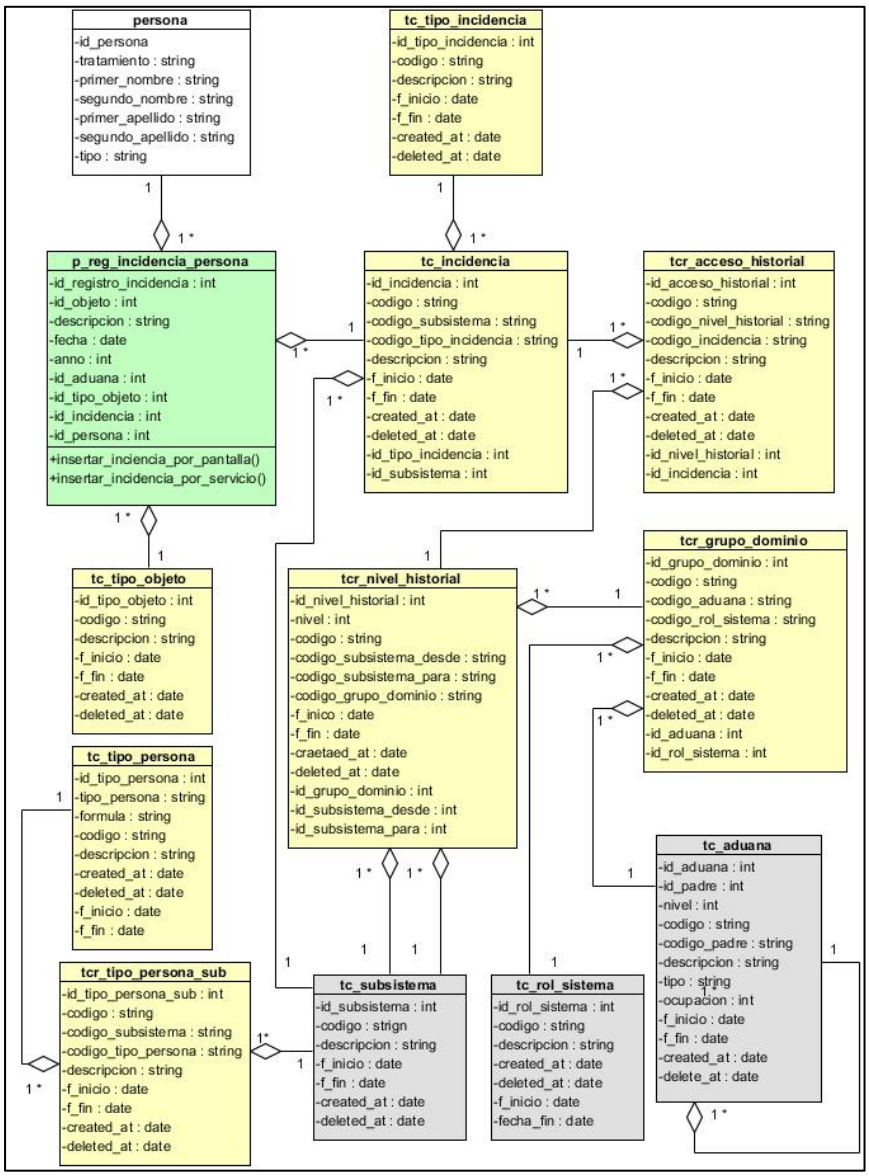


Figura 5. Diagrama de Clases de Negocio.

2.2.1.2 Diagrama de Paquetes

Un diagrama de paquetes es una agrupación de elementos del modelo. Los paquetes se pueden anidar dentro de otros paquetes. Un paquete puede contener paquetes subordinados, así como otros tipos de elementos del modelo. Todo tipo de elementos del modelo UML pueden ser organizados en paquetes. (32)

En la figura 6 se muestra el diagrama de paquetes del componente Historial de Personas Naturales del sistema GINA.

Este diagrama se encuentra compuesto por un paquete principal denominado **sfPersonaPlugin** el cual se encarga de gestionar todo lo referente a la gestión de las personas naturales dentro del sistema GINA. Dicho paquete cuenta con cuatro paquetes internos los cuales se describen a continuación:

- **Web:** encargado de agrupar los ficheros (imágenes, clases CSS y clases JavaScript) que decoran la vista de la aplicación.
- **Lib:** contiene el conjunto de clases del modelo.
- **Config:** contiene los ficheros de configuración destinados al acceso a datos (*databases.yml*) así como el fichero de configuración de los servicios web[11] (*services.yml*).
- **Historial:** es el paquete principal que compone al componente **sfPersona**. Su función es la de centralizar el historial de las personas naturales dentro del sistema GINA a partir de la información brindada por los subsistemas. Constituye la base fundamental sobre la cual se desarrolla la investigación.

Funcionando de forma externa se encuentran las librerías del *framework* Symfony ubicadas en el paquete **Symfony Lib** con el cual interactúan todos los compontes del sistema GINA.

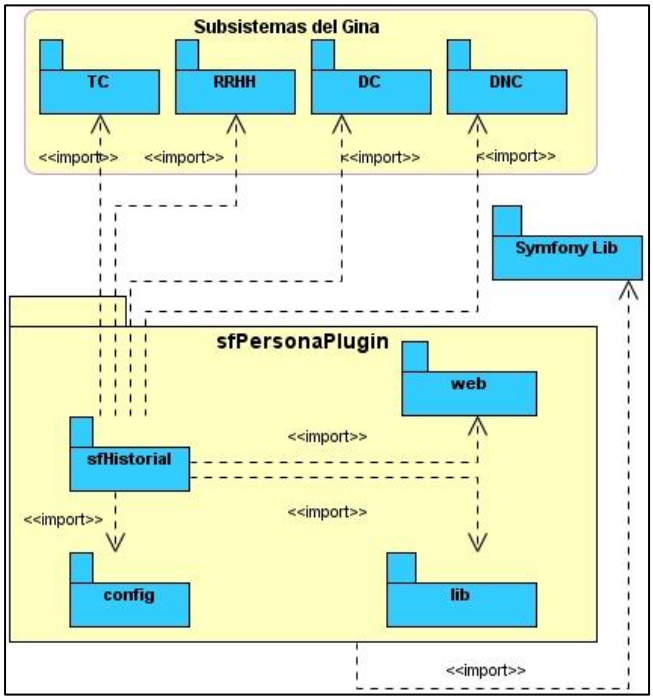


Figura 6. Diagrama de Paquetes del Componente Historial de Personas.

[11] **Servicio Web:** es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

2.2.1.3 Diagramas de interacción

Los diagramas de interacción son modelos que describen la manera en que colaboran grupos de objetos para cierto comportamiento. Estos se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento.

Los diagramas de interacción no son sólo importantes para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables por medio de ingeniería directa e inversa. Un diagrama de interacción es básicamente una proyección de los elementos de una interacción. La semántica del contexto de una interacción, los objetos y roles, enlaces, mensajes y secuenciación se aplican a los diagramas de interacción. Al igual que los demás diagramas, los diagramas de interacción pueden contener notas y restricciones.

En los diagramas de interacción se muestra un patrón de interacción entre objetos. Hay dos tipos de diagrama de interacción, ambos basados en la misma información, pero cada uno enfatizando un aspecto particular: Diagramas de Secuencia y Diagramas de Colaboración. (31)

Diagramas de secuencia orientados a actividades

Estos tipos de diagramas son utilizados en el Departamento de Soluciones para la Aduana. Son elaborados con el propósito fundamental de realizar un diseño más orientado al desarrollo de forma tal que la fase de implementación sea lo más viable, dotando a los programadores de un mayor entendimiento del negocio. Son una combinación de los diagramas de actividades y colaboración propuestos por RUP, permitiendo agregar comentarios para facilitar su entendimiento.

A continuación se explicarán los dos diagramas de este tipo empleados durante el diseño de la presente solución, el primero de ellos orientado al diseño de las actividades del negocio, el otro orientado a la modelación de los componentes visuales.

Diagrama de Secuencia orientado a las Actividades del Negocio

Este tipo de diagrama es el encargado de describir el funcionamiento de cada uno de los Requisitos Funcionales (RF). Dependiendo de la complejidad de cada requisito puede existir más de un diagrama.

En la figura 7 se muestra el diagrama diseñado para el RF Insertar Incidencia por Pantalla. Se compone de ocho calles las cuales se corresponden con las clases del fragmento del negocio que representan.

En este ejemplo se comienza a partir de la entrada de los datos de las personas con sus respectivas incidencias. Posteriormente se envían para ser registrados y son capturados en la clase *action.class.php* del módulo *sfHistorial* el cual verifica que cada persona esté en la base de datos y en caso de no estar la inserta así como sus datos personales (fecha de nacimiento, edad, sexo y nacionalidad). Posteriormente se obtiene el identificador de esta persona y se procede a registrarle la incidencia correspondiente con la llamada a la función *InsertarIncidenciaPorPantalla()* a la cual se le pasa como parámetro los datos de la incidencia además del identificador de la persona.

Tanto en el momento de insertar la persona como en el de registrar la incidencia, el sistema valida que los datos introducidos sean correctos apoyándose para ello en los formularios de validación del *framework* Symfony, en caso de ocurrir algún error se procede a lanzar una excepción indicando el fallo ocurrido.

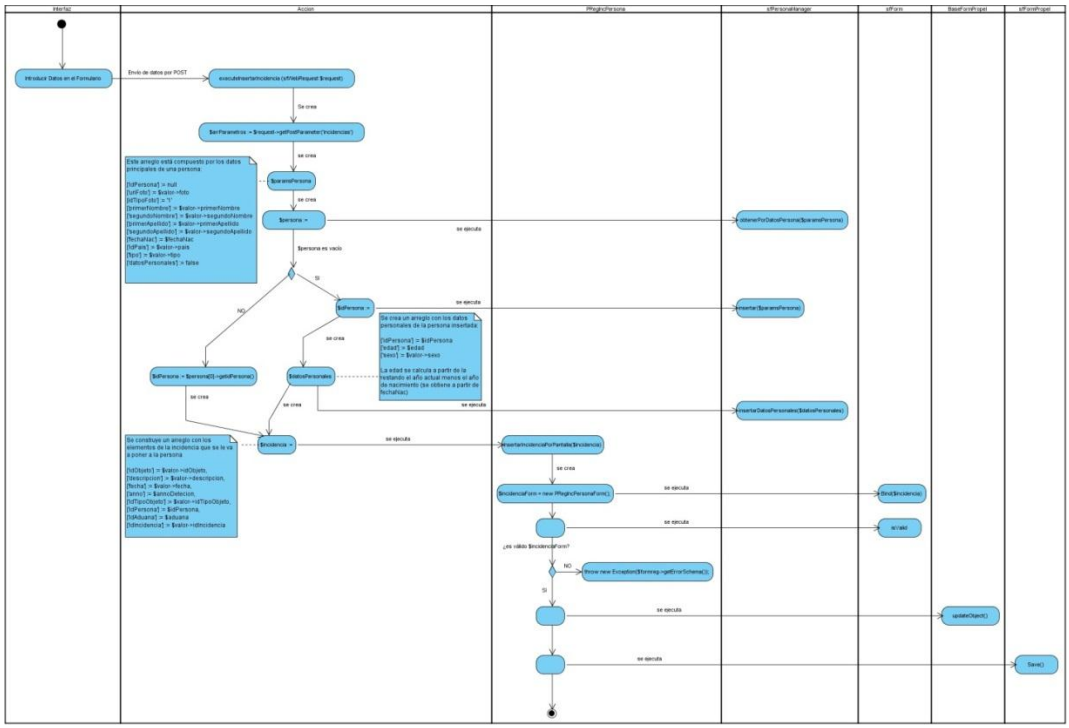


Figura 7. Diagrama de Secuencia orientado a las Actividades del Negocio del RF Insertar Incidencia por Pantalla.

Diagrama de Secuencia orientado a las Actividades de los Componentes Visuales

Este tipo de diagrama tiene como función fundamental dotar a los diseñadores de interfaz de usuario con la información necesaria de forma tal que los procesos de desarrollo se agilicen, al mismo tiempo que se eleve la calidad de los mismos.

Se construye un diagrama para cada uno de los componentes visuales que para su funcionamiento necesitan recibir o enviar algún dato al negocio. En los comentarios de este diagrama se especifican varios parámetros que son útiles para el diseñador como es el caso del evento, nombre de la tabla o funcionalidad de la cual va a cargar o enviar la información entre otros de manera que se garantice una total correspondencia entre la información de este diagrama y su equivalente en el negocio.

En la figura 8 se muestra un ejemplo del tipo de diagrama descrito anteriormente en el cual se muestra el proceso de obtención de los datos personales de un declarante a partir de su identificador. Esta funcionalidad es un complemento del RF Mostrar Historial de Persona.

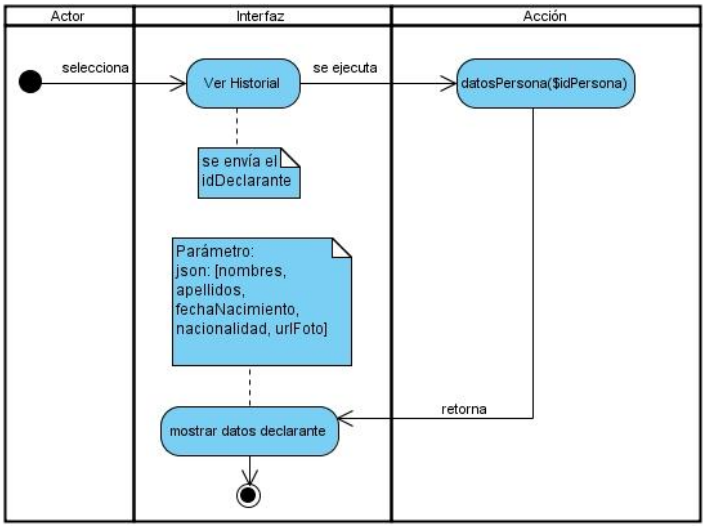


Figura 8. Diagrama de Secuencia orientado a las Actividades de los Componentes Visuales.

En el **Anexo #2** se encuentran otros diagramas que se obtuvieron como parte del diseño de la solución.

2.2.2 Diseño de la Base de Datos

En general, el objetivo del diseño de una base de datos relacional es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información.

El modelo Entidad-Relación (ER) fue propuesto por Peter P. Chen entre los años 1976 - 1977. Posteriormente otros muchos autores han investigado y escrito sobre el modelo, proporcionando importantes aportaciones, por lo que realmente no se puede considerar que exista un único modelo ER.

Dicho modelo describe los datos como entidades, relaciones (vínculos) y atributos y permite representar el esquema conceptual de una base de datos de forma gráfica mediante los diagramas ER.

En la figura 9 se muestra el diagrama ER de la solución. El mismo se encuentra compuesto por 13 tablas cuyos colores representan una ayuda para facilitar el entendimiento de dicho diagrama.

Las representadas con color gris corresponden a las tablas de nomencladores ubicadas fuera del esquema del componente *Persona*. Las tablas representadas con el color amarillo almacenan los nomencladores y se encuentran ubicadas dentro del esquema *Persona*. La tabla representada con el color verde se corresponde con el lugar en el cual se almacenarán los datos de las incidencias cometidas por las personas naturales dentro del sistema GINA. El color blanco representa a la tabla Persona, encargada del almacenamiento de las personas naturales dentro del sistema.

En el **Anexo #1** se puede obtener una descripción de cada tabla, profundizando en cada uno de sus atributos y relaciones.

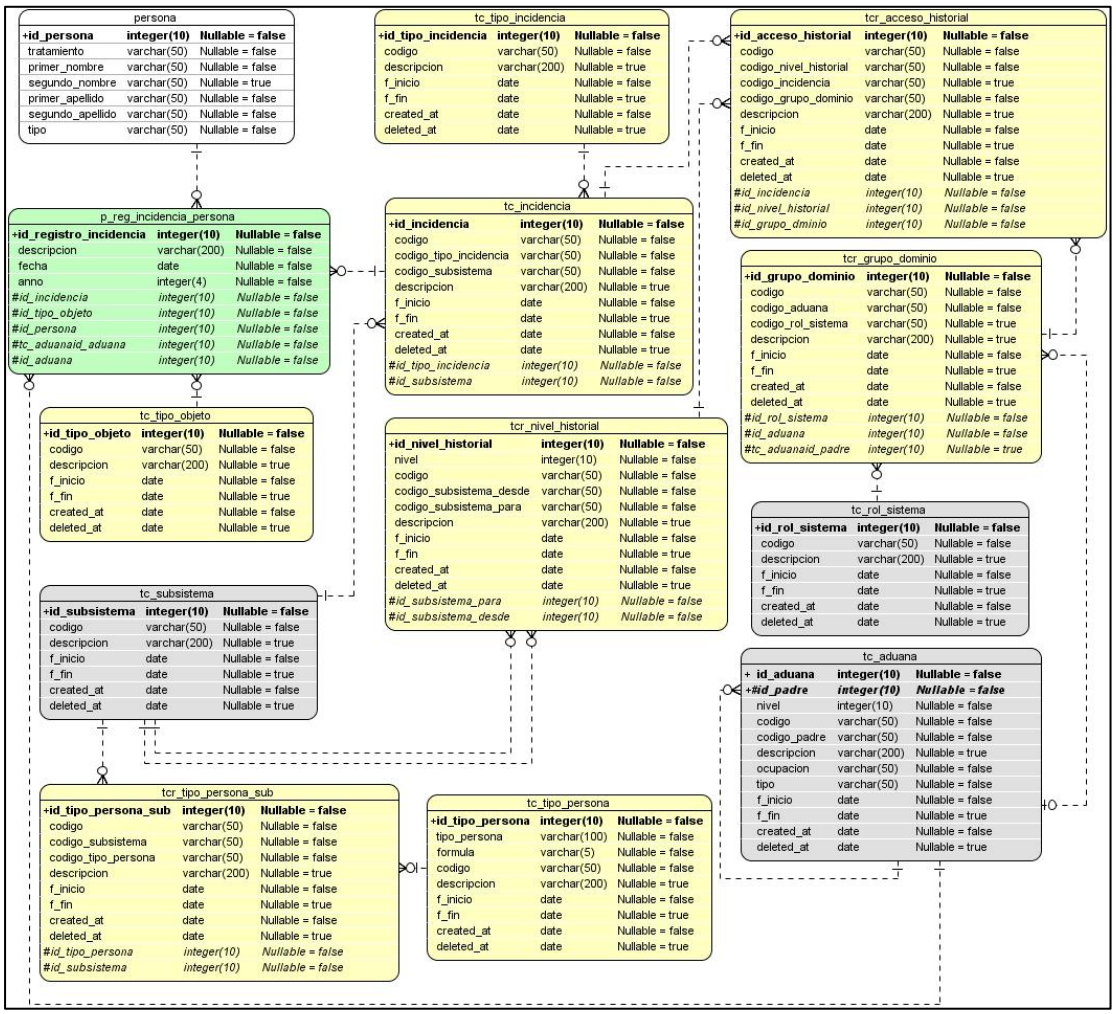


Figura 9. Diagrama Entidad-Relación.

2.3 Implementación del Sistema

Afortunadamente, la mayor parte de la arquitectura del sistema es capturada durante el diseño, siendo el propósito principal de la implementación el desarrollar la arquitectura y el sistema como un todo, de forma más específica, los propósitos de la implementación son:

- Planificar las integraciones del sistema necesarias en cada iteración, siguiendo para ello un enfoque incremental, lo que da lugar a un sistema que se implementa en una sucesión de pasos pequeños y manejables.
- Implementar las clases y subsistemas encontrados durante el diseño. En particular las clases se implementan como componentes de fichero que contienen el código fuente.

- Probar los componentes individualmente, y a continuación integrarlos compilándolos y enlazándolos en uno o más ejecutables, antes de ser enviados para ser integrados y llevar a cabo las comprobaciones del sistema. (31)

2.3.1 Estándares de codificación

Los estándares de codificación son una guía que define la estructura sobre la cual se debe regir el código de la implementación del sistema.

Su objetivo fundamental es que el código fuente esté estructurado de forma común en todo el sistema. Este estándar debe servir de apoyo al personal del proyecto para identificar de forma sencilla cuál es el objetivo y las funcionalidades que brinda cada una de las clases, funciones y demás componentes de software dada su nomenclatura, es necesario que esto se pueda identificar a simple vista. Además debe servir de guía para posteriores implementaciones o modificaciones del sistema.

Para definir el estilo de codificación a seguir en la aplicación se utilizó la notación estándar establecida para aplicaciones desarrolladas en el lenguaje PHP (*PHP Coding Standard*), que mayormente está basada en el estándar de código para aplicaciones en C++ (*C++ Coding Standard*). A continuación se muestran las principales reglas de codificación utilizadas en el desarrollo de la solución, cada regla de estas fue obtenida del documento Propuesta de un Estándar de Codificación del departamento de Soluciones Aduaneras del CEIGE.

2.3.1.1 Reglas Generales

Para la apertura y cierre de las etiquetas del lenguaje PHP será utilizada la siguiente notación:

```
<?php
      //Código
?>
```

Los encabezados en las funciones y acciones deberán contener el requisito a que da solución, los parámetros que necesita así como la respuesta esperada:

```
/**
* RF: Insertar Incidencia
* @params: json
* @return: json
**/
```

2.3.1.2 Reglas en las Acciones

- Dentro de las especificaciones del *framework* Symfony está que cada una de las acciones debe comenzar con la palabra *execute*.

- Todos los nombres de acciones deben estar en la nomenclatura CamelCase[12] comenzando por la palabra execute como son: *executeMostrarHistorial, executeInsertarIncidencia*, etc.
- Los nombres de las acciones deben especificar, con la menor cantidad de palabras, cual es el objetivo de la acción.

2.3.2 Tratamiento de errores

Para garantizar el correcto funcionamiento de cualquier sistema es imprescindible identificar y controlar los posibles errores que se pueden presentar a la hora de interactuar con el software. En el sistema propuesto se tratan estos errores de forma tal que las interacciones con la base de datos (inserción, eliminación, modificación y obtención) se realicen de forma correcta.

Para lograr lo mencionado anteriormente se establecieron mecanismos de validación que comprueben la corrección de los datos a tratar tanto en el lado del cliente como en el lado del servidor.

En el lado del cliente se insistió en que el usuario introduzca la menor cantidad posible de datos, evitando así incoherencias e incorrecciones en los mismos. En el caso de la entrada de datos por parte del usuario se implementaron funciones que validaron dicha entrada para que, de existir errores, se muestren mensajes de alerta indicando los errores detectados.

Para llevar a cabo el control de los datos obtenidos desde el cliente en el servidor se hizo uso de los validadores de formularios introducidos en el *framework* Symfony a partir de su versión 1.2. Esta característica permite que exista una estrecha relación entre los atributos de la base de datos y los datos que se necesiten introducir, haciendo uso de la excepciones del lenguaje PHP en caso de detectarse algún error.

2.3.3 Diagrama de Componentes

Los componentes constituyen la parte modular del sistema, encapsulan implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros, ejecutables, binarios, tablas y documentos). Constituyen recursos desarrollados para un fin concreto y que puede formar solo o junto con otros, un entorno funcional requerido por cualquier proceso predefinido. Son independientes entre ellos, y tienen su propia estructura e implementación. (31)

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre los mismos. Los componentes físicos incluyen archivos, cabeceras,

[12] **CamelCase:** es un estilo de escritura que se aplica a frases o palabras compuestas.

bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

En la figura 10 se muestra el diagrama de componentes obtenido durante el desarrollo de la solución.

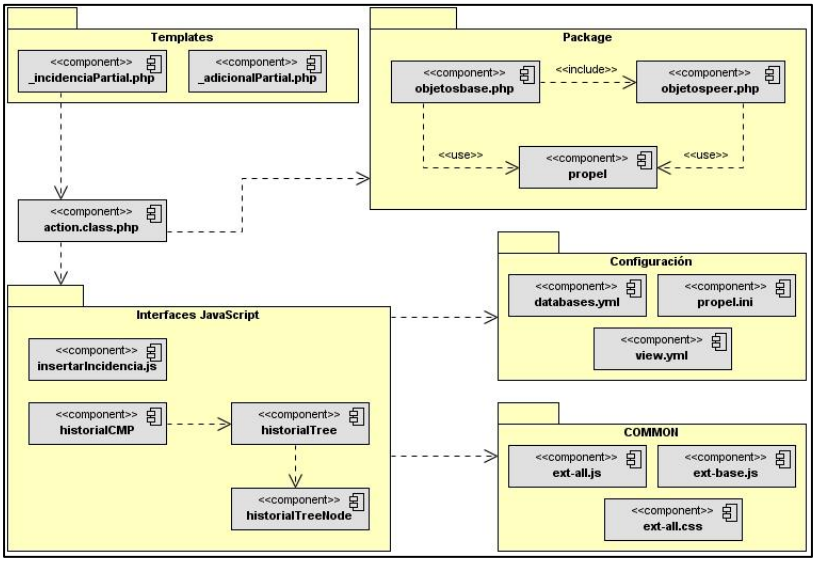


Figura 10. Diagrama de Componentes.

Este diagrama se encuentra compuesto por cinco componentes principales: *Templates*, *Interfaces JavaScript*, *Configuración*, *Package* y *COMMON*; además del componente *Actions* el cual normalmente contiene un único archivo llamado *actions.class.php* y que corresponde a la clase controladora que almacena todas las acciones del módulo correspondiente.

El componente *Configuración* es el encargado de contener los archivos de configuración del acceso a datos *databases.yml* y *propel.ini*; además del fichero de configuración de las vistas *view.yml*. El componente *Interfaces de JavaScript* agrupa las clases de JavaScript encargadas del diseño de las vistas destinadas al registro de las incidencias y la consulta del historial de personas naturales; se comunica directamente con el componente *COMMON* el cual contiene las clases CSS y JavaScript pertenecientes al *framework* Ext JS. El componente *Package* es el encargado de agrupar las clases de configuración del *framework* Symfony. Por último se encuentra el componente *Templates* que agrupa los ficheros *partials* empleados para mostrar detalladamente la descripción del historial de personas naturales.

2.3.4 Comunicación entre las capas

Haciendo uso del patrón arquitectónico MVC, el *framework* Symfony interactúa con las capas del negocio (modelo, vista y controlador) permitiendo una mayor organización del sistema además de incrementar su rendimiento.

En el presente epígrafe se muestran dos ejemplos de comunicación entre las capas del negocio con el objetivo de ayudar a comprender el funcionamiento de la solución. Primeramente se ejemplifica el proceso de registro de personas naturales con las incidencias cometidas por estas; luego el proceso de consulta del historial para una persona natural partiendo de uno de los subsistemas del GINA.

2.3.4.1 Ejemplo #1: Registrar Incidencias

Este proceso comienza luego de que las incidencias sean detectadas por las personas autorizadas y estas vayan a registrarse en el sistema. Para ello el autorizado debe tener los permisos necesarios para poder registrar incidencias a personas naturales. Para acceder a la interfaz se debe hacer a través del subsistema correspondiente al que fue detectada la infracción.

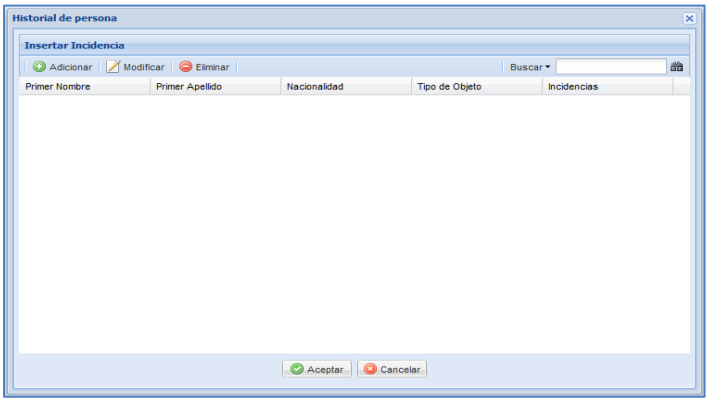


Figura 11. Pantalla principal para registrar las incidencias.

Desde la pantalla que se muestra en la figura 11 se pueden adicionar las personas infractoras pulsando sobre el botón **Adicionar**.

Figura 12. Pantalla para adicionar una persona con la incidencia cometida.

En la pantalla que se muestra en la figura 12 se cargan una serie de datos de forma automática lo cual permite una mayor seguridad de los mismos. Estos datos se cargan de las tablas de nomencladores en las cuales se encuentran definidos aspectos como países y tipos de objetos.

El fragmento de código que se muestra a continuación ejemplifica el proceso de obtención de los datos de la tabla de nomencladores *TC_PAIS* y su posterior ubicación en un componente de tipo *tcCombo*[13] para mostrar la nacionalidad de la persona.

```
/* Fragmento de código perteneciente al llenado de un componente ComboBox haciendo uso del componente tcCombo */

{
    fieldLabel: 'Nacionalidad',
    xtype: 'tcCombo',
    gridColumn: false,
    tabla: 'tc_pais',
    name: 'pais'
}
```

Para ser utilizado en otras tablas sólo es necesario modificarle los parámetros de configuración: *fieldLabel*, *table* y *name*. De la misma forma son cargados los datos correspondientes al Tipo de Objeto.

Para la obtención de los datos correspondientes a los campos Tipo Persona e Incidencias, fue necesario tener en cuenta que sólo se deberían cargar aquellos datos que tuvieran relación con el subsistema desde el cual se va a registrar la incidencia. Para ello fue necesario implementar dos funcionalidades, una para cada caso:

[13] **tcCombo:** es un componente visual para el *framework* Ext JS desarrollado en el Departamento de Soluciones para la Aduana cuyo propósito es la obtención de los nomencladores a partir del nombre de la tabla correspondiente.

```
/* Fragmentos de código correspondientes a la obtención de las incidencia correspondientes al subsistema activo */
// plugins/sfPersonaPlugin/modules/sfHistorial/actions/action.class.php

        $criteria = new Criteria();
        $criteria->add(TcIncidenciaPeer::ID_SUBSISTEMA, $idSubsistema);
        $array = array('nombreTc' => TcIncidenciaPeer::TABLE_NAME,
          'criteria' => $criteria,
          'estado' => SisTcTablaPeer::VIGENTE);
        $component = SysComponentsLocator::getInstance();
        $tcComponent = $component->getComponent('tc');
        $incidencias = $tcComponent->executeService(new sfEvent(null, 'tc.obtenerDadoCriteria', $array));
        $arrayIncidencias = array();
        foreach ($incidencias as $valor) {
          $arrayTemporal['id'] = $valor->getIdIncidencia();
          $arrayTemporal['texto'] = $valor->getDescripcion();
          $arrayIncidencias[] = $arrayTemporal;
        }
        $result['incidencias'] = $arrayIncidencias;

        return $this->renderText(json_encode($result));
      } catch (PropelException $e) {
        throw new SysExceptions($e->getMessage(), 500);
      }
```

```
/* Fragmentos de código correspondientes a la obtención de los tipos de personas correspondientes al subsistema activo */
// plugins/sfPersonaPlugin/modules/sfHistorial/actions/action.class.php

        $criteria = new Criteria();
        $criteria->add(TcrTipoPersonaSubPeer::ID_SUBSISTEMA, $idSubsistema);
        $criteria->addJoin(TcrTipoPersonaSubPeer::ID_TIPO_PERSONA, TcTipoPersonaPeer::ID_TIPO_PERSONA);
        $array = array('nombreTc' => TcTipoPersonaPeer::TABLE_NAME,
          'criteria' => $criteria,
          'estado' => SisTcTablaPeer::VIGENTE);

        $component = SysComponentsLocator::getInstance();
        $tcComponent = $component->getComponent('tc');
        $tiposPersona = $tcComponent->executeService(new sfEvent(null, 'tc.obtenerDadoCriteria', $array));

        $arrayTipos = array();
        foreach ($tiposPersona as $valor) {
          $arrayTemporal['id'] = $valor->getIdTipoPersona();
          $arrayTemporal['texto'] = $valor->getDescripcion();
          $arrayTipos[] = $arrayTemporal;
        }
        $result['tipos'] = $arrayTipos;

        return $this->renderText(json_encode($result));
      } catch (PropelException $e) {
        throw new SysExceptions($e->getMessage(), 500);
      }
```

Como se puede apreciar, en ambos casos se consume un servicio web correspondiente al subsistema de Tablas de Control (*obtenerDadoCriteria*). Este servicio web tiene como objetivo la obtención de un objeto a partir de la ejecución de un criterio dado.

Si al adicionar una incidencia no son introducidos los datos correctamente el sistema lanzará un mensaje de error indicándole al usuario los campos que debe corregir, similar a como se muestra en la figura 13.

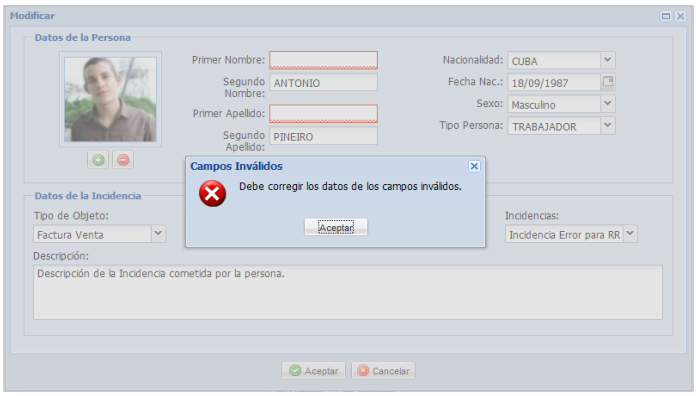


Figura 13. Mensaje de error indicando que existen campos incorrectos.

Cuando todos los campos se encuentren llenos correctamente el usuario pulsará sobre el botón **Aceptar** de la figura 13. Todos estos datos pasarán a un estado de espera como se muestra en la figura 14, dando la posibilidad de adicionar todas las incidencias detectadas.

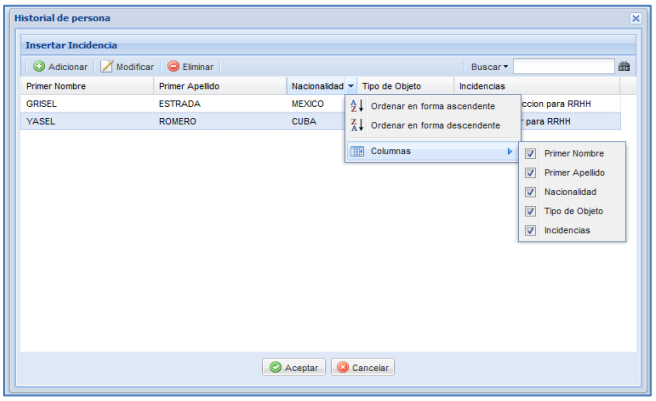


Figura 14. Incidencias en espera de ser guardadas.

Cuando todas las incidencias han sido registradas se procederá a guardarlas en la base de datos pulsando sobre el botón **Aceptar**. (Figura 14).

El siguiente fragmento de código corresponde a la clase ***insertarIncidencia.js*** y en el mismo se muestra la función que se encarga de enviar los datos introducidos hacia la clase controladora ***actions.class.php*** para ser guardados posteriormente.

```
// plugins/sfPersonaPlugin/web/js/sfHistorial/ InsertarIncidencia.js
```

```
function() {
        if (obj.form.isValid() && Ext.getCmp('idIncidencia').getGrid().getStore().getCount() != 0) {
          obj.form.submit({
            url: 'sfHistorial/insertarIncidencia',
            waitMsg: 'Enviando...',
            waitTitle: 'Espere por favor',
            success: function(form, action){
              var msg = Ext.decode(action.response.responseText);
              msg = "La solicitud se ha guardado con el n&uacute;mero " + msg.numeroSolicitud;
              Ext.Msg.show ({
                title: obj.title,
                msg: msg,
                icon: Ext.Msg.INFO,
                buttons: Ext.Msg.OK,
                width: 350
              });
              Ext.getCmp ('mainPanelExt').getLayout().setActiveItem(0);
            },
            failure: FAILURE_SUBMIT
          });
        }
        else{
          Ext.Msg.show ({
            title: 'Error',
            msg: 'Existen campos con datos inv&aacute;lidos.',
            icon: Ext.Msg.ERROR,
            buttons: Ext.Msg.OK,
            width: 300
          });
        }
}
```

La línea número 6 indica que esta función enviará los datos hacia una acción nombrada ***insertarIncidencia()*** que se encuentra dentro del módulo ***sfHistorial***, parte de este fragmento de código se muestra a continuación.

```
// plugins/sfPersonaPlugin/modules/sfHistorial/actions/action.class.php

  public function executeInsertarIncidencia(sfWebRequest $request) {
    try {
      $con = Propel::getConnection('persona');
      $con->beginTransaction();

      $incidencias = json_decode($request->getPostParameter('Incidencia'));

      foreach ($incidencias as $valor) {
        // Conformar la fecha de la persona
        $fechaSinFormato = (explode('T', $valor->fechaNac));
        $fechaNac = $fechaSinFormato[0];
        $anno = explode('-', $fechaNac);
        $edad = date('Y') - $anno[0];

        // Conformar el arreglo con los parametros de la persona
        $params = array(
```

```
            'idPersona' => NULL,
            'urlFoto' => $valor->foto,
            'idTipoFoto' => '1',
            'primerNombre' => $valor->primerNombre,
            'segundoNombre' => $valor->segundoNombre,
            'primerApellido' => $valor->primerApellido,
            'segundoApellido' => $valor->segundoApellido,
            'fechaNac' => $fechaNac,
            'idPais' => $valor->pais,
            'tipo' => $valor->tipo,
            'datosPersonales'=> false
        );

        // Buscar la persona
        $persona  = sfPersonaManager::obtenerPorDatosPersona(new sfEvent(null, null, $params));

        // Comprobar que existe
        if (!empty($persona)){
            $idPersona  = $persona[0]->getIdPersona();
        } else {
            // inserto la persona
            $idPersona = sfPersonaManager::insertar(new sfEvent(null, null, $params));

            unset($params['urlFoto']);
            unset($params['idTipoFoto']);
            unset($params['primerNombre']);
            unset($params['segundoNombre']);
            unset($params['primerApellido']);
            unset($params['segundoApellido']);
            unset($params['tipo']);
            unset($params['datosPersonales']);

            // inserto los datos personales
            $params['edad'] = $edad;
            $params['idPersona'] = $idPersona;
            $params['sexo'] = $valor->sexo;
            sfPersonaManager::insertarDatosPersonales(new sfEvent(null, null, $params));
        }

        //Contruir los datos de la incidencia
        $fechaSinFormato = (explode('T', $valor->fecha));
        $fechaIncidencia = $fechaSinFormato[0];
        $annoIncidencia = explode('-', $fechaIncidencia);
        $incidencia = array(
            'idObjeto' => $valor->idObjeto,
            'descripcion' => $valor->descripcion,
            'fecha' => $fechaIncidencia,
            'anno' => $annoIncidencia[0],
            'idTipoObjeto' => $valor->idTipoObjeto,
            'idPersona' => $idPersona,
            'idAduana' => 2,
            'idIncidencia' => $valor->idIncidencia
        );

        // Insertar la incidencia para la persona actual
        PRegIncPersona::InsertarIncidenciaPorPantalla($incidencia);        }        $con->commit();
    return $this->renderText('{success:true}');
```

```
    } catch (Exception $e) {
      throw new Exception($e->getMessage());
      $con->rollBack();
    }
  }
```

En el fragmento de código anterior se realizan dos pasos importantes:

- **Buscar persona**: se realiza la búsqueda de las personas a las cuales se le va a registrar la incidencia. En caso de no encontrarse se devuelve un objeto vacío, al contrario se devuelve el identificador de la persona.

Cuando la persona no es encontrada se procede a registrarla a partir de sus datos personales.

```
$idPersona = sfPersonaManager::insertar(new sfEvent(null, null, $params));
```

- **Insertar Incidencia**: se le registran incidencias a las personas con el identificador devuelto en el paso descrito anteriormente.

```
// plugins/sfPersonaPlugin/lib/model/PRegIncPersona.php

public static function InsertarIncidenciaPorPantalla($param)  {
    $formreg = new PRegIncPersonaForm();
    // if($request->isMethod('post')){
    $formreg->bind($param);
    if ($formreg->isValid()) {
      $formreg->updateObject();
      $formreg->save();
    } else {
      $error = $formreg->getErrorSchema();
      throw new Exception($error);
    }
  }
```

Si todo procede sin anormalidades se mostrará un mensaje indicando que el proceso ha terminado satisfactoriamente (Figura 15).

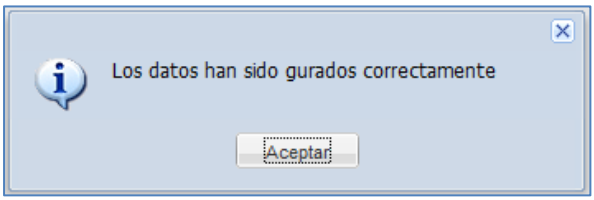


Figura 15. Mensaje indicando que los datos se han guardado de forma correcta.

En caso contrario se mostrará una alerta indicando que el proceso no ha podido ejecutarse.

2.3.4.2 Ejemplo #2: Consultar Historial

Este proceso comienza a partir de que se tiene un conjunto de personas en un determinado subsistema y el usuario a cargo del sistema desea consultar el historial de una de estas personas.

Este ejemplo se enmarca en el módulo de ***Solicitudes*** perteneciente al subsistema ***Despacho Comercial***. En el listado que se muestra en la figura 16 se observan las personas naturales que se encuentran pendientes a ***Gestionar Solicitud de Facilidad***.

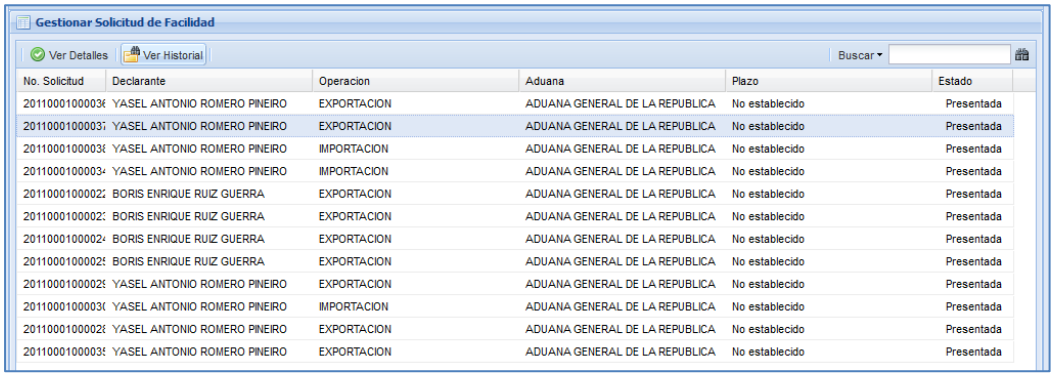


Figura 16. Personas pendientes a Gestionar Facilidad.

Para acceder al historial de una de las personas del listado anterior primeramente se selecciona dicha persona y posteriormente se ejecuta el botón **Ver Historial** cuyo código se muestra a continuación.

```
// web/js/dc/solicitud/GestionarFacilidadCotejo.js

{
        text: 'Ver Historial',
        iconCls: 'iconFolderFind',
        handler: function(){
                var record = obj.getSelectionModel().getSelected();
          if (record != null) {
             new Ext.Window({
                 title: 'Historial de persona',
                 width: 750,
                 height: 480,
                 layout: 'fit',
                 maximizable: true,
                 items: {
                    xtype: 'persona.historial',
                    idPersona: record.data.idDeclarante
                 }
              }).show();
          }
          Else {
             Ext.Msg.show({
                title: obj.title,
                width: 300,
                msg: 'Seleccione un elemento.',
                buttons: Ext.Msg.OK,
                icon: Ext.MessageBox.ERROR
             });
          }
        }
    }
```

La ejecución del código anterior envía el identificador de la persona a la cual se le solicitó ver el historial hacia la clase de JavaScript ***historial.js*** y esta a su vez ejecuta dos funcionalidades implementadas en la clase controladora ***actions.class.php***: ***executeDatosPersona()***, encargada de mostrar los datos personales de la persona en cuestión y ***executeObtenerHistorial()*** encargada de la obtención de forma organizada del historial de la persona. (Figura 17).

El fragmento de código que se muestra a continuación retorna los datos personales mencionados anteriormente.

```
// plugins/sfPersonaPlugin/modules/sfHistorial/actions/action.class.php

/**
   * Action DatosPersona
   * @param $idPersona = el id de la persona que se quiere buscar
   * @return JSON con los datos de la persona
   */
  public function executeDatosPersona(sfWebRequest $request) {
    try {
       $idpersona = $request->getPostParameter('idPersona');
       $datosPersona = PersonaPeer::DatosPersona($idpersona);

       $json = "{success: true, data: {";
       $json .= "nombre:" . "'" . $datosPersona['nombre'] . "',";
       $json .= "apellido:" . "'" . $datosPersona['apellido'] . "',";
       $json .= "fechaNacimiento:" . "'" . $datosPersona['fechaNacimiento'] . "',";
       $json .= "nacionalidad:" . "'" . $datosPersona['nacionalidad'] . "',";
       $json .= "image:" . "'" . $datosPersona['image'] . "'";
       $json .= "}}";

       return $this->renderText($json);
    } catch (PropelException $e) {
       throw new SysExceptions($e->getMessage(), 500);
    }
  }
```

El historial de las personas se organiza de dos grupos fundamentales:

- **Historial de Incidencias:** agrupa el historial de las incidencias cometidas pero las personas naturales en los diferentes procesos de la aduana.
- **Historial Adicional:** agrupa el historial que no es de incidencias. Este es brindado por cada subsistema al componente ***Historial de Personas*** a través de un servicio web nombrado ***obtenerHistorial*** el cual debe ser implementado internamente por cada subsistema.

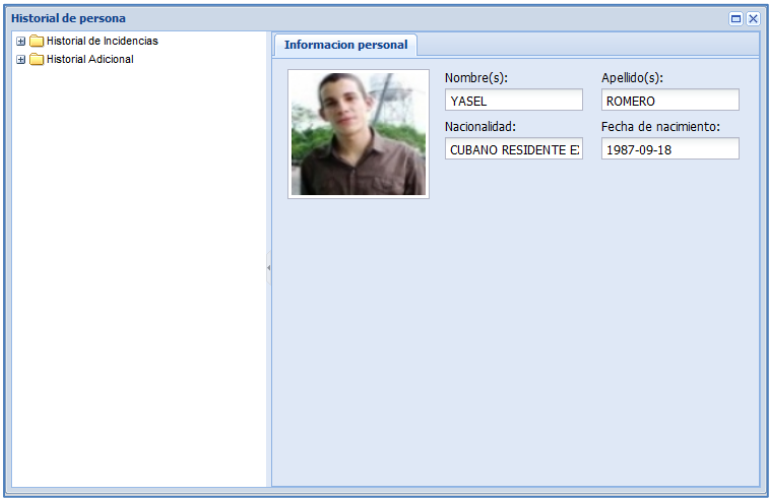


Figura 17. Ventana principal del historial para una persona dada.

El fragmento de código que se presenta a continuación muestra como se procede a la hora de realizar el llenado del árbol de exploración del historial situado a la izquierda de la pantalla que se mostró en la figura 17.

```
// plugins/sfPersonaPlugin/modules/sfHistorial/actions/action.class.php

switch ($tipoHistorial) {
    // Inicialmente se muestran las dos opciones de historiales
    case "null":{
                        return $this->renderText("[{
                         id: '1-0',
                         text: 'Historial de Incidencias',
                         tipoHistorial: 'incidencia',
                         codigo: 'null',
                         anno: 'null',
                         tipoIncidencia: 'null',
                         fecha: 'null',
                         leaf: false
                },{
                         id: '2-1',
                         text: 'Historial Adicional',
                         tipoHistorial: 'adicional',
                         codigo: 'null',
                         anno: 'null',
                         fecha: 'null',
                         leaf: false
                }]");
               }
               break;

case "incidencia":{
     // Código para el historial de las incidencias
        }
        break;
```

```
case "adicional": {
    // Código para el historial adicional
        }
        break;
```

Como se puede apreciar, la funcionalidad está organizada mediante una sentencia ***switch()*** compuesta por tres bloques. El primero de ellos (***null***) se emplea para cuando se ejecuta la pantalla inicial debido a que no se conoce qué tipo de historial desea consultar el usuario. Los dos bloques restantes (***incidencia*** y ***adicional***) se ejecutan en dependencia del tipo de historial que el usuario seleccione. (Figura 18).

Cada bloque actualiza los atributos que va a retornar de forma escalonada hasta llegar a la descripción del historial, ya sea de la incidencia o adicional, en ese caso el valor *leaf*, el cual indica si se está o no al final del árbol, actualiza su valor a *true* y se realiza la llamada a la funcionalidad *executeMostrarHistorial()*.

```
// plugins/sfPersonaPlugin/modules/sfHistorial/actions/action.class.php

/**
   * Accion que retorna una incidencia dado el idRegistroIncidencia
   * @param sfWebRequest $request
   * @return renderPartial
   */
  public function executeMostrarHistorial(sfWebRequest $request) {
    $arrParams = explode("-", $request->getPostParameter('id'));

    switch ($arrParams[1]) {
      case 'inc': {
          $idRegistroIncidencia = $arrParams[0];
          $registroIncidencia = PRegIncPersonaPeer::retrieveByPK($idRegistroIncidencia);

          $syscomponent = SysComponentsLocator::getInstance();
          $componente = $syscomponent->getComponent('tc');

          //Obteniendo la aduana
          $params = array('nombreTc' => TcAduanaPeer::TABLE_NAME, 'id' => $registroIncidencia->getIdAduana());
          $aduana = $componente->executeService(new sfEvent(null, 'tc.obtenerRegistroXid', $params));

          //Obteniendo el Tipo de Objeto
          $params = array('nombreTc' => TcTipoObjetoPeer::TABLE_NAME, 'id' => $registroIncidencia->getIdTipoObjeto());
          $tipoObjeto = $componente->executeService(new sfEvent(null, 'tc.obtenerRegistroXid', $params));

          //Obteniendo el Tipo de Incidencia
          $params = array('nombreTc' => TcIncidenciaPeer::TABLE_NAME, 'id' => $registroIncidencia->getIdIncidencia());
          $incidencia = $componente->executeService(new sfEvent(null, 'tc.obtenerRegistroXid', $params));
          $params = array('nombreTc' => TcTipoIncidenciaPeer::TABLE_NAME, 'id' => $incidencia->getIdTipoIncidencia());
          $tipoIncidencia = $componente->executeService(new sfEvent(null, 'tc.obtenerRegistroXid', $params));

          return $this->renderPartial('incidenciasPartial', array(
            'descripcion' => $registroIncidencia->getDescripcion(),
            'aduana' => $aduana->getDescripcion(),
```

```
                'codigo' => $aduana->getCodigo(),
                'tipoIncidencia' => $tipoIncidencia->getDescripcion(),
                'objeto' => $tipoObjeto->getDescripcion(),
                'identificador' => $registroIncidencia->getIdObjeto(),
                'fecha' => $registroIncidencia->getFecha('d-m-Y'),
                'nombreCompleto' => $registroIncidencia->getPersona()->getNombreCompleto()
            ));
        }
        break;

    case 'adic': {
        return $this->renderPartial('adicionalPartial', array(
            'descripcion' => $arrParams[2]
        ));
        }
        break;
    }
}
```

La funcionalidad mostrada anteriormente es la encargada de mostrar una descripción más completa del historial seleccionado como se puede apreciar en la figura 18.

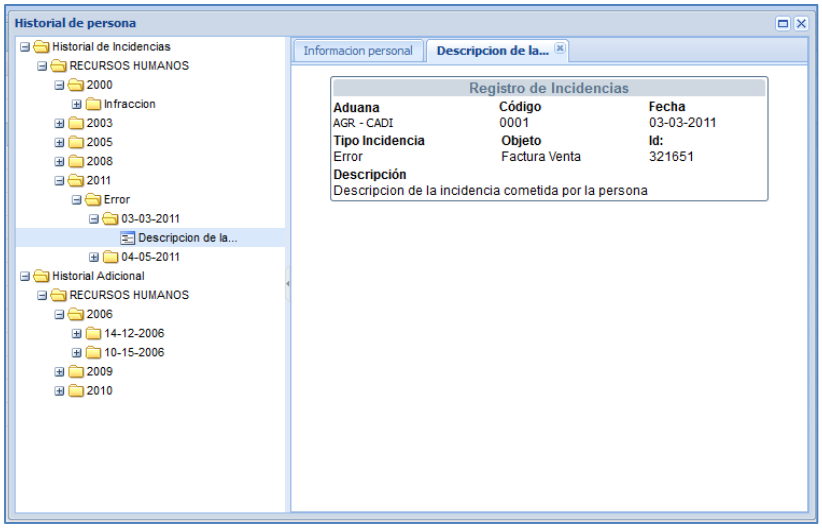


Figura 18. Pantalla con la descripción detallada del historial de una persona natural.

En caso de seleccionarse el historial de incidencias la descripción contará con la aduana en la que se cometió, la fecha de detección y el tipo de incidencia, entre otros aspectos. En caso de tratarse del historial adicional la descripción estará dada por los elementos enviados desde el subsistema donde se registró el historial.

Implementación y configuración del servicio web *mostrarHistorial*

El servicio web ***mostrarHistorial***, como se mencionó anteriormente, debe ser implementado por cada subsistema si este necesita mostrar información del historial adicional de una persona natural. El

fragmento de código que se muestra a continuación ejemplifica las sentencias básicas para implementar una funcionalidad que muestre de forma organizada el historial adicional. (Figura 19).

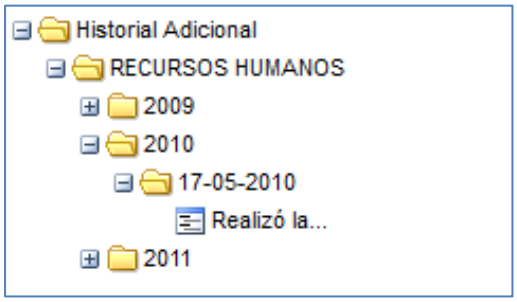


Figura 19: Organización del árbol de Historial Adicional

```
/*Fragmento de código que muestra la implementación de una variante que resuelva la funcionalidad de mostrar
el historial adicional en cada subsistema*/

static public function obtenerHistorial(sfEvent $event) {
    $arregloParametros = $event->getParameters();
    $idPersona = $arregloParametros['idPersona'];
    $nivelAcceso = $arregloParametros['nivel'];
    $arregloCriterios = $arregloParametros['criterios'];

    switch ($arregloCriterios['criterio']) {
            case 'anno':
            //Código aquí
            // Se retorna un arreglo con los año
                    break;

            case 'fecha':
            //Código aquí
            // Se retorna un arreglo con los fechas para el año seleccionado
                    break;

            case 'descripcion':
            //Código aquí
            // Se retorna un arreglo con las descripciones del historial en la fecha        selecionada
                    break;
    }
}
```

Para lograr un correcto funcionamiento del servicio web implementado en cada subsistema se debe proceder a registrarlo, de forma que el *framework* Symfony lo reconozca y sea capaz de ejecutarlo correctamente. Este proceso es sencillo, para ello sólo es necesario actualizar la configuración del fichero ***services.yml*** del subsistema correspondiente. El ejemplo que se muestra a continuación corresponde a la actualización del servicio en el subsistema Recursos Humanos.

```
obtenerHistorial:
  class: RhFicha
  method: obtenerHistorial
  desc: Obtiene el historial adicional de una persona natural
```

Esta configuración consta de cuatro parámetros: nombre del servicio, clase en la que se implementó la funcionalidad, nombre de la funcionalidad y descripción de la misma.

2.3.5 Resumen del proceso de implementación

La etapa de implementación resultó una de las más complejas en el desarrollo de la solución. Se obtuvieron un total de 16 funcionalidades distribuidas en las distintas clases del negocio.

Se hizo uso del componte *sfUtilPlugin* el cual posibilitó la interacción a través de servicios web con los subsistemas del GINA. Se consumieron un total de seis servicios web, los cuales son llamados en 25 ocasiones durante los procesos de registro de incidencias y consulta del historial.

Para lograr una estandarización en la comunicación entre la solución obtenida y los subsistemas del GINA se hizo necesaria la implementación de un servicio web en cada subsistema, el cual es consumido por el componente Historial de Personas Naturales en tres ocasiones cada vez que se consulta el historial adicional de una persona.

Para lograr que los subsistemas registraran las incidencias de forma automática se hizo necesario la implementación de un servicio web en el componente Historial de Personas Naturales el cual es consumido una vez por cada incidencia que necesite registrar un determinado subsistema.

2.4 Conclusiones parciales

La obtención de los principales artefactos durante la etapa de diseño posibilitó una ejecución más viable de la implementación del sistema.

Se evidenciaron las potencialidades del *framework* Symfony para potenciar el desarrollo de aplicaciones web dinámicas de gestión.

Capitulo 3. Validación de la solución propuesta

3.1 Introducción

Las pruebas constituyen el único instrumento adecuado para determinar el status de la calidad de un producto de software. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que este cumple con los requerimientos. Básicamente los objetivos fundamentales del proceso de pruebas se describen a continuación:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.
- Diseñar e implementar las pruebas creando los casos de pruebas que especifican qué probar, creando los procedimientos de prueba que especifican cómo realizarlas y creando, si es posible, componentes de pruebas ejecutables para automatizarlas.
- Realizar las diferentes pruebas y manejar los resultados de cada una sistemáticamente. Las construcciones en las que se detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que las no conformidades puedan ser corregidas. (31)

En este capítulo se mostrará el proceso de pruebas realizado al componente Historial de Personas Naturales, argumentando inicialmente en las diferentes técnicas de pruebas de software que existen.

3.2 Técnicas de Evaluación de Software

La calidad de un sistema está determinada, entre otras cosas, por la coincidencia entre lo que se concibió y los requisitos establecidos en la primera fase. Para comprobar el grado de cumplimiento de estos requisitos se usan dos técnicas de pruebas de software las cuales se describen a continuación:

Técnicas de Evaluación Estáticas: buscan faltas sobre el sistema en reposo. Estudian los distintos modelos que componen el sistema de software buscando posibles faltas en los mismos. Así pues, estas técnicas se pueden aplicar, tanto a requisitos como a modelos de análisis, diseño y código.

Técnicas de Evaluación Dinámicas: generan entradas al sistema con el objetivo de detectar fallos cuando el sistema ejecuta dichas entradas. Los fallos se observan cuando se detectan incongruencias

entre la salida esperada y la salida real. La aplicación de técnicas dinámicas es también conocida como pruebas de software o *testing* y se aplican generalmente sobre código. (33)

Como se indicó anteriormente, las técnicas de evaluación dinámica proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. Estas técnicas se agrupan en:

- **Técnicas de caja blanca o estructurales**, que se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- **Técnicas de caja negra o funcionales**, que realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. (33)

3.3 Estrategias de pruebas de Caja Negra aplicadas

Las estrategias de pruebas de Caja Negra consisten en pasos a seguir a la hora de evaluar dinámicamente un sistema de software, comenzando por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto.

3.3.1 Pruebas Unitarias

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

Las pruebas unitarias del *framework* Symfony son archivos PHP normales cuyo nombre termina en *Test.php* y que se encuentran en el directorio *test/unit/* de la aplicación. Su sintaxis es sencilla y fácil de implementar.

3.3.1.1 Aplicación de pruebas unitarias al sistema

A continuación se muestra el diseño de casos de prueba para el RF **Consultar Historial de Incidencias**, nótese que se encuentra compuesto por tres escenarios los cuales agrupan cada una de las funcionalidades que dieron solución a dicho requisito.

Tabla 1: Diseño de casos de prueba para el RF Consultar Historial de Incidencias

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario

Consultar Historial de Incidencias.	Permite consultar el historial de incidencias de una persona natural dada.	EP 1.1: Mostrar datos personales.	• Se muestra una interfaz con los datos personales de la persona a consultar el historial: nombres, apellidos, fecha de nacimiento, nacionalidad y foto.
		EP 1.2: Mostrar árbol con el historial de incidencias.	• Se muestra un árbol con dos opciones, la primera debe corresponder con **Historial de Incidencias.** • Al desplegar esta opción se muestran los subsistemas en los cuales la persona tiene historial de incidencias ordenados alfabéticamente. • Al seleccionar un subsistema se muestran los años en los cuales la persona tiene historial ordenados de menor a mayor. • Al seleccionar un año se muestran los tipos de incidencias ordenados alfabéticamente. • Al seleccionar un tipo de incidencia se muestran las fechas correspondientes a las incidencias ordenadas de menor a mayor. • Al seleccionar una fecha se muestran las descripciones de las incidencias resumidas en menos de 20 caracteres ordenados alfabéticamente.
		EP 1.3: Mostrar descripción avanzada del registro de incidencias	• Al seleccionar la descripción resumida de un registro de incidencia se muestra una pestaña con la descripción avanzada del registro.

A continuación se muestran las pruebas unitarias del *framework* Symfony aplicadas a algunos de los escenarios descritos anteriormente.

EP1.1: Mostrar datos personales

Esta funcionalidad debe retornar, dado el identificador de una persona, un arreglo de cinco posiciones con los datos personales de la persona correspondiente a dicho identificador.

En el caso que se muestra a continuación se tomó la persona con identificador dos (*$idPersona = 2*), la comparación de los resultados se muestran en la siguiente tabla.

Tabla 2: resultados esperados y obtenidos al aplicar las pruebas unitarias del *framework* Symfony al escenario de pruebas 1.1.

No.	Descripción	Resultado esperado	Resultado obtenido
1	Tipo de resultado	array	array
2	Nombre de la persona	YASEL	YASEL
3	Primer apellido	ROMERO	ROMERO
4	Fecha de nacimiento	1987-09-18	1987-09-18
5	Nacionalidad	CUBANO RESIDENTE EXTERIOR	CUBANO RESIDENTE EXTERIOR
6	URL de la foto	../uploads/fotosCI/yasel.jpg	../uploads/fotosCI/yasel.jpg

Las pruebas unitarias del *framework* Symfony para el caso anterior se definieron de la siguiente forma:

```
$idPersona = 2;
$result = DatosPersona($idPersona);

$t->isa_ok($result, 'array');
$t->is($result['nombre'], 'YASEL');
$t->is($result['apellido'], 'ROMERO');
$t->is($result['fechaNacimiento'], '1987-09-18');
$t->is($result['nacionalidad'], 'CUBANO RESIDENTE EXTERIOR');
$t->is($result['image'], '../uploads/fotosCI/yasel.jpg');
```

El ejecutar estas pruebas el resultado arrojado es el siguiente:

```
www\GINA>symfony test:unit HistorialPersona
1..6
ok 1
ok 2
ok 3
ok 4
ok 5
ok 6
Looks like everything went fine.
```

El resultado anterior demuestra que las seis pruebas realizadas se ejecutaron correctamente.

EP 1.2: Mostrar árbol con el historial de incidencias

El proceso de pruebas para este escenario resulta un poco complicado debido a que la información que se muestra es organizada en forma de árbol, por lo que hay que probar cada nivel de este como se describió en la tabla del diseño del caso de pruebas.

A continuación se muestra el proceso de las pruebas unitarias del *framework* Symfony aplicadas al escenario de pruebas 1.2, específicamente al nivel dos del árbol el cual es el encargado de mostrar los

subsistemas del GINA en los que la persona dada tiene historial de incidencias. Para este caso se toma la misma persona del ejemplo anterior (*$idPersona = 2*) y la tabla que se muestra a continuación describe la comparación de los resultados.

Tabla 3: resultados esperados y obtenidos al aplicar las pruebas unitarias del *framework* Symfony al escenario de pruebas 1.2.

No.	Descripción	Resultado esperado	Resultado obtenido
1	Tipo de resultado	array	array
2	Cantidad de resultados	1	1
3	Nombre del subsistema	RECURSOS HUMANOS	RECURSOS HUMANOS
4	Código del subsistema	RRHH	RRHH
5	Tipo de historial	incidencia	incidencia
6	Año de la incidencia	null	null
7	Tipo de la incidencia	null	null
8	Fecha de la incidencia	null	null
9	Es hoja del árbol	false	false

La implementación en el *framework* Symfony de las pruebas unitarias para este ejemplo se muestra a continuación:

```
$test = new lime_test(9, new lime_output_color());
$idPersona = 2;
$subsistemas = subsistemas($idPersona);

$test->isa_ok($subsistemas, 'array');
$test->is(count($subsistemas), 1);
$test->is($subsistemas[0]['text'], 'RECURSOS HUMANOS');
$test->is($subsistemas[0]['codigo'], 'RRHH');
$test->is($subsistemas[0]['tipoHistorial'], 'incidencia');
$test->is($subsistemas[0]['anno'], 'null');
$test->is($subsistemas[0]['tipoIncidencia'], 'null');
$test->is($subsistemas[0]['fecha'], 'null');
$test->is($subsistemas[0]['leaf'], false);
```

Al ejecutar estas pruebas los resultados obtenidos son los esperados:

```
www/GINA>symfony test:unit HistorialPersona
1..9
ok 1
ok 2
ok 3
ok 4
ok 5
ok 6
```

```
ok 7
ok 8
ok 9
 Looks like everything went fine.
```

Como se pudo apreciar en los dos ejemplos mostrados anteriormente, las pruebas unitarias arrojaron los resultados esperados. Cabe destacar que cada línea prueba una única cosa, de esa forma es posible detectar mayor cantidad de errores, lo que posibilita que cada funcionalidad pueda ser evaluada con un número elevado de pruebas así como con diferentes datos.

De la misma forma se le realizaron las pruebas al resto de los escenarios obteniéndose, en la mayoría de los casos, los resultados propuestos.

3.3.2 Pruebas Funcionales

Las pruebas funcionales constituyen una estrategia de evaluación de software. El objetivo fundamental de estas pruebas es validar si el comportamiento observado del software cumple o no con sus especificaciones definidas. La prueba funcional toma el punto de vista del usuario. Las funciones son probadas ingresando las entradas y examinando las salidas. La estructura interna del programa raramente es considerada.

3.3.2.1 Aplicación de pruebas funcionales al sistema

A continuación se muestra el diseño de caso de pruebas para el RF **Insertar Incidencia por Pantalla**. Se encuentra distribuido en cinco escenarios de pruebas, de forma que las funcionalidades sean probadas con un mayor número de juego de datos.

Tabla 4: Diseño de casos de prueba para el RF Insertar Incidencia por Pantalla.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Insertar Incidencia por pantalla	Permite que se inserte una incidencia acompañada de la persona infractora.	EP 1.1: Insertar incidencias.	• Selecciona la opción del menú Insertar Incidencia. • Se muestra una interfaz desde la cual se pueden adicionar, eliminar y modificar las incidencias antes de ser guardadas.
		EP 1.2: Adicionar Incidencias.	• Selecciona la opción **Adicionar**. • Se muestra una interfaz desde la cual es posible adicionar incidencias así como a la persona incidente para ser insertadas posteriormente en la base de datos. • Si los datos son correctos pasarán hacia la pantalla del EP 1.1 para luego ser

			guardados.
		EP 1.3: Modificar incidencias.	• Selecciona la opción **Modificar**. • Se muestra una interfaz desde la cual es posible modificar las incidencias así como a la persona incidente antes de ser insertadas posteriormente en la base de datos. • Si los datos son correctos pasarán hacia la pantalla del EP 1.1 para luego ser guardados.
		EP 1.4: Eliminar incidencias.	• Selecciona una incidencia de las que están en la lista para ser guardadas. • Selecciona la opción **Eliminar**. • Se muestra un mensaje de alerta preguntando si se desea eliminar el registro. • Si se confirma este mensaje la incidencia es eliminada del listado.
		EP 1.5: Guardar incidencias.	• Selecciona la opción **Guardar**. • Se muestra un mensaje de información indicando que las incidencias se han guardado satisfactoriamente. • Si no existen incidencias para guardar u ocurrió algún error, se notifica a través de una alerta.

Los escenarios de pruebas descritos anteriormente fueron probados con los juegos de datos que se muestran en la tabla 5.[14]

Tabla 5: Juego de datos para el RF Insertar Incidencia por Pantalla.

Id EP	EP	Foto	1er Nomb	2do Nomb	1er Apell	2do Apell	F. Nac	Nacionalidad	Sexo	T. Per	T. Obj	Número	F. Det	Inc	Desc	Resp. sistema	Res. prueba
1.1	Insertar incidencias	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Muestra una pantalla con las opciones iniciales para el registro de las incidencias.	Se muestra la pantalla con las opciones iniciales.

[14] Las celdas de la tabla contienen **V**, **I**, o **NA**. **V** indica válido, **I** indica inválido, y **NA** no se debe proporcionar un valor del dato en este caso.

1.2	Adicionar Incidencias.	V	V	V	V	V	V	V	V	V	V	V	V	V	V	Almacena los datos de la incidencia en la ventana principal.	Se guardan de forma temporal las incidencias en una tabla.
		V	I	V	V	I	V	V	V	I	V	V	I	V	V	Muestra un mensaje indicando que existen datos incorrectos y sombrea dichos campos	Se muestra un mensaje indicando que hay campos incorrectos. **No se sombrea el campo con el 1er nombre y el 1er apellido.**
1.3	Modificar incidencias	V	V	V	V	V	V	V	V	V	V	V	V	V	V	Se carga una pantalla con los datos temporales a modificar.	Se muestra una interfaz con los datos a modificar.
		V	V	V	V	V	V	V	V	I	I	V	V	V	V	Muestra un mensaje indicando que existen datos incorrectos y sombrea dichos campos.	Los campos Tipo Objeto y Tipo de Persona son marcados como incorrectos.
1.4	Eliminar incidencias	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Se muestra una alerta. En caso de confirmarla se elimina la incidencia.	Se elimina la incidencia de la lista.
1.5	Guardar incidencias.	V	V	V	V	V	V	V	V	V	V	V	V	V	V	Se muestra un mensaje indicando que las incidencias han sido guardadas.	Las incidencias son guardadas en la base de datos.

		NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Se muestra un mensaje indicando que no existen incidencias para guardar.	Mensaje de alerta indicando que no existen incidencias para guardar.

Como se pudo apreciar, en la segunda prueba que se le aplicó al escenario de prueba 1.2 se detectaron dos no conformidades. La esencia de estas consiste en que los campos de texto primer nombre y segundo apellido permitieron la entrada de caracteres no válidos, dando la posibilidad de que el actor pueda formular un valor incorrecto.

3.3.3 Resumen del proceso de validación

Las pruebas de Caja Negra fueron aplicadas al resto de los RF de la misma forma que se describió en el epígrafe anterior. Los resultados por escenarios para cada una de las funcionalidades se describen a continuación:

Funcionalidades en Insertar Incidencia:

- Insertar Incidencia por pantalla: 5 escenarios. 2 no conformidades.
- Insertar Incidencia por servicio. 3 escenarios.
- Obtener Tipos de Personas por subsistemas: 3 escenarios.
- Obtener Tipos de Incidencias por subsistemas: 3 escenarios.

Funcionalidades en Consultar Historial de Personas Naturales:

- Consultar historial de incidencias: 3 escenarios.
- Consultar historial adicional: 5 escenarios.
- Mostrar descripción avanzada del historial de incidencias: 2 escenarios.
- Mostrar descripción avanzada del historial adicional: 2 escenarios.

De forma general, las funcionalidades fueron descompuestas en 26 escenarios, detectándose dos no conformidades para un 8 porciento en la primera iteración. (Figura 19).

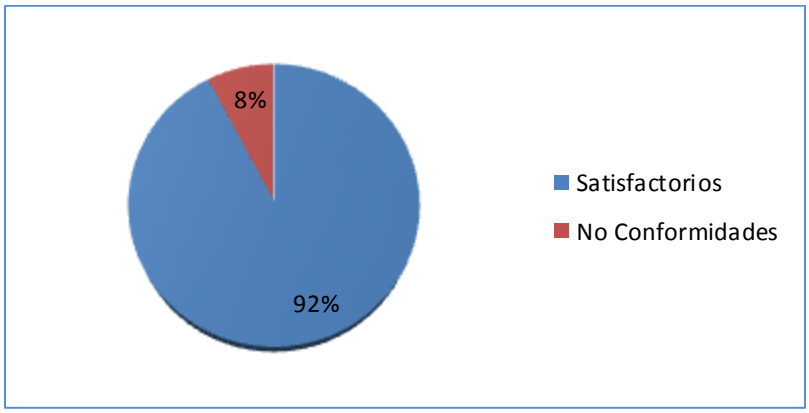


Figura 20: Resultados de las pruebas

Las no conformidades detectadas durante la primera iteración fueron corregidas completamente, lo que posibilitó que en la segunda iteración los escenarios arrojaran resultados satisfactorios en todos los casos. Las pruebas fueron ejecutadas por el grupo de analistas del subsistema Despacho Comercial. (Ver anexo 4).

3.4 Aporte y novedad de la solución

Con la obtención de la solución propuesta, se introdujo una herramienta para la gestión del historial de personas naturales en la AGR nunca antes implementada, contribuyendo de esta forma a la independencia tecnológica del país y al proceso de informatización de la sociedad cubana.

Haciendo uso de herramientas y tecnologías libres, en su gran mayoría, fue posible obtener una solución que es capaz de integrarse al sistema GINA, permitiendo la interacción con los subsistemas concluidos, y adaptable como para integrarse a los subsistemas que serán implementados en un futuro.

Las impresiones de los especialistas del CADI, al valorar la herramienta obtenida, indican que el tiempo de consulta del historial de una persona natural se reduce considerablemente, incrementado el número de personas consultadas en un 80 porciento aproximadamente. Otro de los grandes beneficios atribuidos es el hecho de que el historial se encontrará almacenado en formato digital y ejecutado sobre una tecnología web, posibilitando el acceso a este desde varios puntos del territorio nacional.

3.5 Conclusiones parciales

El estudio de las Técnicas de Evaluación de Software permitió que se profundizara en el uso de las pruebas de Caja Negra aplicadas a proyectos de software.

Capítulo 3. Validación de la solución propuesta

La aplicación de pruebas de Caja Negra al componente obtenido posibilitó la detección de algunas no conformidades, permitiendo que estas fueran corregidas totalmente, dando paso a que el sistema se encuentre totalmente funcional.

Conclusiones

El estudio de un conjunto de herramientas y tecnologías así como su posterior empleo posibilitó la obtención de la solución propuesta inicialmente, cumpliendo con las tareas y objetivos propuestos.

El componente obtenido durante la investigación permitió la centralización del Historial de Personas Naturales dentro del sistema GINA, posibilitando que la consulta del histórico de una determinada persona natural pueda realizarse en el menor tiempo posible y obteniendo los mejores resultados.

Recomendaciones

- Incorporar el componente obtenido a la gestión del historial de personas naturales en los futuros subsistemas del sistema GINA.
- Extender las funcionalidades de gestión de historial a las personas jurídicas.
- Continuar con el seguimiento de las actualizaciones de las herramientas y tecnologías informáticas empleadas en la solución para garantizar mejoras en futuras versiones del sistema.

Referencias Bibliográficas

1. *Aduana General de la República de Cuba.* [En línea] 2008. [Citado el: 8 de Febrero de 2011.] http://www.aduana.co.cu/.

2. *ADUANA SIGLO XXI.* **D, Lic. LUIS ML. SÁNCHEZ.** Santo Domingo, D.N : s.n., 2007.

3. **Fierro, A.** *El patrimonio en las sociedades comerciales. Aplicaciones juridicas y contables.* s.l. : ECOE EDICIONES, 2002. 9789586482981.

4. *Revista Pedagogía Universitaria.* **García., Dr. C. María Elena Guardo.** 3, Facultad Cultura Física. Matanzas. : s.n., 2009, Vol. XIV.

5. **Berndtsson, Mikael, y otros, y otros.** *Thesis Projects: A Guide for Students in Computer Science.* Springer-Verlag London : s.n., 2002,. ISBN-13: 978-1-84800-008-7.

6. **Real Academia Española.** *Diccionario.* [En línea] [Citado el: 8 de Febrero de 2011.] http://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&TIPO_BUS=3&LEMA=aduana.

7. **Ramírez, Arturo.** *Las Aduanas.* [En línea] Enero de 2005. [Citado el: 10 de Febrero de 2011.] http://www.aduanas.com.ve/boletines/boletin_12/aduana.htm.

8. **Real Academia Española.** *Diccionario.* [En línea] [Citado el: 8 de Febrero de 2011.] http://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&TIPO_BUS=3&LEMA=historial.

9. **Arellano, Ana Alejandra Febres.** RRPP/ UNCTAD-SIDUNEA. *Conociendo al SIDUNEA - Sistema Aduanero Automatizado.* [En línea] Marzo de 2004. [Citado el: 16 de Febrero de 2011.] http://www.gestiopolis.com/canales2/economia/sidunea.htm.

10. **Korea Customs Service.** *Customs Automation.* [En línea] 2006. [Citado el: 8 de Febrero de 2011.] http://english.customs.go.kr/.

11. **DDS.** *DDS: Sistemas Informáticos.* [En línea] 2010. [Citado el: 8 de Febrero de 2011.] http://www.ddsoftware.com.ar/software-comercio-exterior.html.

12. **Grupo Soluciones Innova.** *Rational Unified Process.* [En línea] [Citado el: 8 de Febrero de 2011.] http://www.rational.com.ar/herramientas/rup.html.

Referencias Bibliográficas

13. **Reyero, Eusebio.** *Metodologías de diseño | Will Web For Food.* [En línea] 2008. [Citado el: 6 de Febrero de 2011.] http://wwff.thespacer.net/blog/metodologias-de-diseno/.

14. **Larman, Craig.** *UML y Patrones.* 1999. ISBN 970-1 7-0261-1.

15. **Fowler, Martin.** *Patterns of Enterprise Application Architecture.* ISBN: 0-32112-742-0.

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

17. **Universidad Tecnológica Nacional - Facultad Regional Tucumán .** *Lenguajes de Programación.* [En línea] 2000. [Citado el: 22 de Febrero de 2011.] http://frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm.

18. **Pérez, Javier Eguíluz.** *Introducción a XHTML.* 2007.

19. —. *Introducción a CSS.* 2007.

20. **Pixelkit.** *Diccionario Web - Estudio de Contenidos Digitales.* [En línea] 2009. [Citado el: 2 de Febrero de 2011.] http://pixelkit.cl/diccionario-web/.

21. **Universidad Carlos III de Madrid.** *JavaScript.* [En línea] 2006. [Citado el: 2 de Febrero de 2011.] http://perso.wanadoo.es/javascript_12/.

22. **PHP.net.** PHP: Hypertext Preprocessor. *PHP: General Information.* [En línea] 2001. [Citado el: 20 de Febrero de 2011.] http://www.php.net/manual/en/faq.general.php.

23. Programación en Castellano. *¿Por qué elegir PHP?* [En línea] 1998. [Citado el: 22 de Febrero de 2011.] http://www.programacion.com/articulo/por_que_elegir_php_143.

24. **Rodriguez, Jose Antonio Cobo.** *Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas.* 2008.

25. **CodeBox.** *Glosario.* [En línea] 2008. [Citado el: 4 de Febrero de 2011.] http://www.codebox.es/glosario.

26. **Comunidad de Desarrollo de Ext JS.** *Ext JS en Español.* [En línea] [Citado el: 4 de Febrero de 2011.] http://extjs.es/.

Referencias Bibliográficas

27. **Escuela de Ingeniería Informática Universidad de Oviedo.** *Entornos de Desarrollo Integrado.* [En línea] 2008. [Citado el: 6 de Febrero de 2011.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%de%Desarrollo%Integrado.pdf.

28. **Oracle Corporation.** *Portal del IDE Java de Código Abierto.* [En línea] 2010. [Citado el: 6 de Febrero de 2011.] http://netbeans.org/index_es.html.

29. **Visual Paradigm International.** *UML, BPMN and Database Tool for Software Development.* [En línea] [Citado el: 8 de Febrero de 2011.] http://www.visual-paradigm.com/.

30. **CAVSI.com.** *¿Qué es un Sistema Gestor de Bases de Datos o SGBD?* [En línea] 2007. [Citado el: 6 de Febrero de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

31. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 1999. ISBN: 0-201-57169-2.

32. *OMG Unified Modeling Language.* 1999.

33. **Juristo, Natalia, Moreno, Ana M. y Vegas, Sira.** *Técnicas de Evaluación de Software.* 2006.

Bibliografía Consultada

- **Collin, Peter.** Publishing Dictionary of Computing, 4th Edition.
- **Conallen, Jim**. 1999. Modeling Web Application Architectures with UML. 1999.
- **Markiewicz, Marcus Eduardo y de Lucena, Carlos J.P.** El Desarrollo del Framework Orientado al Objeto.
- **Meléndrez, Edelsys Hernández**. 2006. Cómo escribir una tesis. La Habana: s.n., 2006.
- **S., Carlos Mario Vélez**. 2005. Apuntes de metodología de la investigación. Medellín: s.n., 2005.
- **Potencier, Fabien. 2009.** *El tutorial Jobeet.* 2009.

Glosario de Términos

Aduana: es una oficina pública de constitución fiscal establecida generalmente en costas y fronteras.

AGR: Aduana General de la República de Cuba.

AJAX: *Asynchronous JavaScript And XML* o JavaScript asíncrono y XML, es una técnica de desarrollo web para crear aplicaciones interactivas. Esta se ejecuta en el lado de cliente y mantiene comunicación asíncrona con el servidor en segundo plano.

AOLServer: es el servidor web de código abierto de América *Online*. Tiene procesamiento multihilo, y se usa para sitios web dinámicos de gran tamaño.

CSS: *Cascade Style Sheet* u Hojas de Estilos en Cascada es un lenguaje creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML.

DHTML: *Dynamic HTML.* Designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de los lenguajes HTML y CSS así como la jerarquía de objetos del DOM.

DOM: *Document Object Model* es un modelo computacional a través de la cual los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML.

Ext JS: es una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de navegadores que nos permite crear páginas e interfaces web dinámicas.

HTML: *HyperText Markup Language*. Es el lenguaje de marcado predominante para la elaboración de páginas web.

IDE: Entorno de Desarrollo Integrado. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

IIS: *Internet Information Services* o IIS es un servidor web y un conjunto de servicios para el sistema operativo *Microsoft Windows*.

Java: es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

Glosario de Términos

JavaScript: es un lenguaje de programación interpretado utilizado principalmente en la realización páginas web. Presenta una sintaxis semejante a la del lenguaje Java y el lenguaje C.

JSON: *JavaScript Object Notation* es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

LGPL: Licencia Pública General Reducida es una licencia de software creada por la *Free Software Foundation.*

Linux: GNU/Linux es uno de los términos empleados para referirse a la combinación del núcleo o kernel libre similar a Unix denominado Linux, que es usado con herramientas de sistema GNU.

Oracle: es un sistema de gestión de base de datos objeto-relacional desarrollado por Oracle Corporation.

PDO: *PHP Data Objects* es una extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.

PHP: *PHP Hypertext Pre-processor* es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

phpDocumentor: consiste en un estándar formal para comentar código PHP.

Roxen: es un servidor web de código abierto con todas las funciones y se distribuye bajo la licencia GPL. Funciona en diferentes sistemas operativos incluyendo Windows, Linux, Solaris y Mac OS X.

RUP: *Rational Unified Process*, es un proceso de desarrollo de software, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

SGML: son las siglas de Standard *Generalized Markup Language*. Consiste en un sistema para la organización y etiquetado de documentos.

Symfony: *framework* de trabajo para la realización de aplicaciones web escrito en PHP 5.

THTTPD: es un servidor web de código libre disponible para la mayoría de las variantes de Unix. Se caracteriza por ser simple, pequeño, portátil, rápido, y seguro, ya que utiliza los requerimientos mínimos de un servidor HTTP.

Glosario de Términos

UCI: Universidad de las Ciencias Informáticas.

UML: es el Lenguaje de Modelado Unificado para detallar, construir, visualizar y documentar las partes o artefactos de un software.

UNIX: es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T.

XML: *eXtensible Markup Language*, es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C)

Windows: es el nombre de la familia de sistemas operativos desarrollados por Microsoft desde 1981.

Anexos

Anexo #1: Descripción del modelo de datos

Nombre: p_reg_incidencia_persona		
Descripción: Tabla encargada del almacenamiento de los registros correspondientes a las incidencias cometidas por las personas naturales.		
Atributo	**Tipo**	**Descripción**
id_registro_incidencia	sequence	Identificador del registro
descripcion	varchar	Descripción de la incidencia
fecha	date	Fecha de detección de la incidencia
anno	integer	Año de detección de la incidencia
id_incidencia	integer	Identificador de la incidencia a que pertenece
id_tipo_objeto	integer	Identificador del objeto sobre el cual se cometió la incidencia
id_persona	integer	Identificador de la persona que cometió la incidencia
id_aduana	integer	Identificador de la aduana desde la cual se detectó la incidencia

Nombre: tc_tipo_objeto		
Descripción: Tabla de Control encargada del almacenamiento de los tipos de objetos.		
Atributo	**Tipo**	**Descripción**
id_tipo_objeto	sequence	Identificador del tipo de objeto
codigo	varchar	Código del tipo de objeto
descripcion	varchar	Descripción del tipo de objeto
f_inicio	date	Fecha de inicio del objeto
f_fin	date	Fecha fin del objeto
created_at	date	Fecha de creación del objeto en la base de datos
deleted_at	date	Fecha de eliminación del objeto en la base de datos

Nombre: tc_tipo_incidencia
Descripción: Tabla de Control encargada del almacenamiento de los tipos de

incidencias.		
Atributo	**Tipo**	**Descripción**
id_tipo_incidencia	sequence	Identificador del tipo de incidencia
codigo	varchar	Código del tipo de incidencia
descripcion	varchar	Descripción del tipo de incidencia
f_inicio	date	Fecha de inicio del tipo de incidencia
f_fin	date	Fecha fin del tipo de incidencia
created_at	date	Fecha de creación del tipo de incidencia en la base de datos
deleted_at	date	Fecha de eliminación del tipo de incidencia en la base de datos

Nombre: tc_incidencia		
Descripción: Tabla de Control encargada del almacenamiento de las incidencias que pueden ser cometidas por las personas naturales así como el subsistema desde donde se cometió.		
Atributo	**Tipo**	**Descripción**
id_tipo_incidencia	sequence	Identificador de la incidencia
codigo	varchar	Código de la incidencia
codigo_tipo_incidencia	varchar	Código del tipo de incidencia
codigo_subsistema	varchar	Código del subsistema
descripcion	varchar	Descripción de la incidencia
f_inicio	date	Fecha de inicio de la incidencia
f_fin	date	Fecha fin de la incidencia
created_at	date	Fecha de creación de la incidencia
deleted_at	date	Fecha de eliminación de la incidencia
id_tipo_incidencia	integer	Identificador del tipo de incidencia
id_subsistema	integer	Identificador del subsistema

Nombre: tcr_grupo_dominio		
Descripción: Tabla de Control encargada del almacenamiento de las relaciones a nivel de base de datos de las aduanas y los roles del sistema.		
Atributo	**Tipo**	**Descripción**
id_grupo_dominio	sequence	Identificador del grupo-dominio

código	integer	Código del grupo-dominio
codigo_aduana	varchar	Código de la aduana de la aduana a que corresponde.
codigo_rol_sistema	varchar	Código del rol de sistema a que corresponde
descripcion	varchar	Descripción del grupo-dominio
f_inicio	date	Fecha de inicio del grupo-dominio
f_fin	date	Fecha fin del grupo-dominio
created_at	date	Fecha de creación del grupo-dominio
deleted_at	date	Fecha de eliminación del grupo-dominio
id_rol_sistema	integer	Identificador del rol de sistema a que corresponde
id_aduana	integer	Identificador de la aduana a que corresponde

Nombre: tcr_nivel_historial		
Descripción: Tabla de Control encargada del almacenamiento del los datos relacionados entre los subsistemas y los grupos-dominios. A cada una de estas relaciones le asigna un nivel el cual será usado para consultar el historial de las personas naturales.		
Atributo	**Tipo**	**Descripción**
id_nivel_historial	sequence	Identificador del nivel-historial
nivel	integer	Nivel de acceso al historial
codigo	varchar	Código del nivel
codigo_subsistema_desde	varchar	Código del subsistema desde el cual se a acceder al historial
codigo_subsistema_para	varchar	Código del subsistema al cual se va a consultar el historial
descripcion	varchar	Descripción del nivel
f_inicio	date	Fecha de inicio del nivel
f_fin	date	Fecha fin del nivel
created_at	date	Fecha de creación del nivel
deleted_at	date	Fecha de eliminación del nivel
id_ subsistema_desde	integer	Identificador del subsistema desde el cual se a acceder al historial
id_ subsistema_para	integer	Identificador del subsistema al cual se va a consultar el historial

Nombre: tcr_acceso_historial		
Descripción: Tabla de control encargada del almacenamiento de los niveles de historiales que pueden acceder a las incidencias.		
Atributo	**Tipo**	**Descripción**
id_acceso_historial	sequence	Identificador del acceso al historial
código	integer	Código del acceso al historial
codigo_nivel_historial	varchar	Código del nivel-historial
codigo_incidencia	varchar	Código de la incidencia
descripcion	varchar	Descripción del acceso al historial
f_inicio	date	Fecha de inicio del acceso al historial
f_fin	date	Fecha fin del acceso al historial
created_at	date	Fecha de creación del acceso al historial
deleted_at	date	Fecha de eliminación del acceso al historial
id_incidencia	integer	Identificador de la incidencia
id_nivel_historial	integer	Identificador del nivel-historial

Nombre: tc_tipo_persona		
Descripción: Tabla de Control encargada del almacenamiento de los tipos de personas que se manejan en la AGR.		
Atributo	**Tipo**	**Descripción**
id_tipo_persona	sequence	Identificador del tipo de persona
codigo	varchar	Código del tipo de persona
descripcion	varchar	Descripción del tipo de persona
f_inicio	date	Fecha de inicio del tipo de persona
f_fin	date	Fecha fin del tipo de persona
created_at	date	Fecha de creación del tipo de persona en la base de datos
deleted_at	date	Fecha de eliminación del tipo de persona en la base de datos

Nombre: tcr_tipo_persona_sub		
Descripción: Tabla de Control encargada del almacenamiento de los tipos de personas que son manejados en cada subsistema.		
Atributo	**Tipo**	**Descripción**

id_tipo_persona_sub	sequence	Identificador del tipo de persona
codigo	varchar	Código del tipo de persona-subsistema
codigo_subsistema	varchar	Código del subsistema
codigo_tipo_persona	varchar	Código del tipo de persona
descripcion	varchar	Descripción del tipo de persona-subsistema
f_inicio	date	Fecha de inicio del tipo de persona-subsistema
f_fin	date	Fecha fin del tipo de persona-subsistema
created_at	date	Fecha de creación del tipo de persona-subsistema en la base de datos
deleted_at	date	Fecha de eliminación del tipo de persona-subsistema en la base de datos
id_subsistema	integer	Identificador del subsistema
id_tipo_persona	integer	Identificador del tipo de persona

Anexo #2: Diagramas de Secuencia

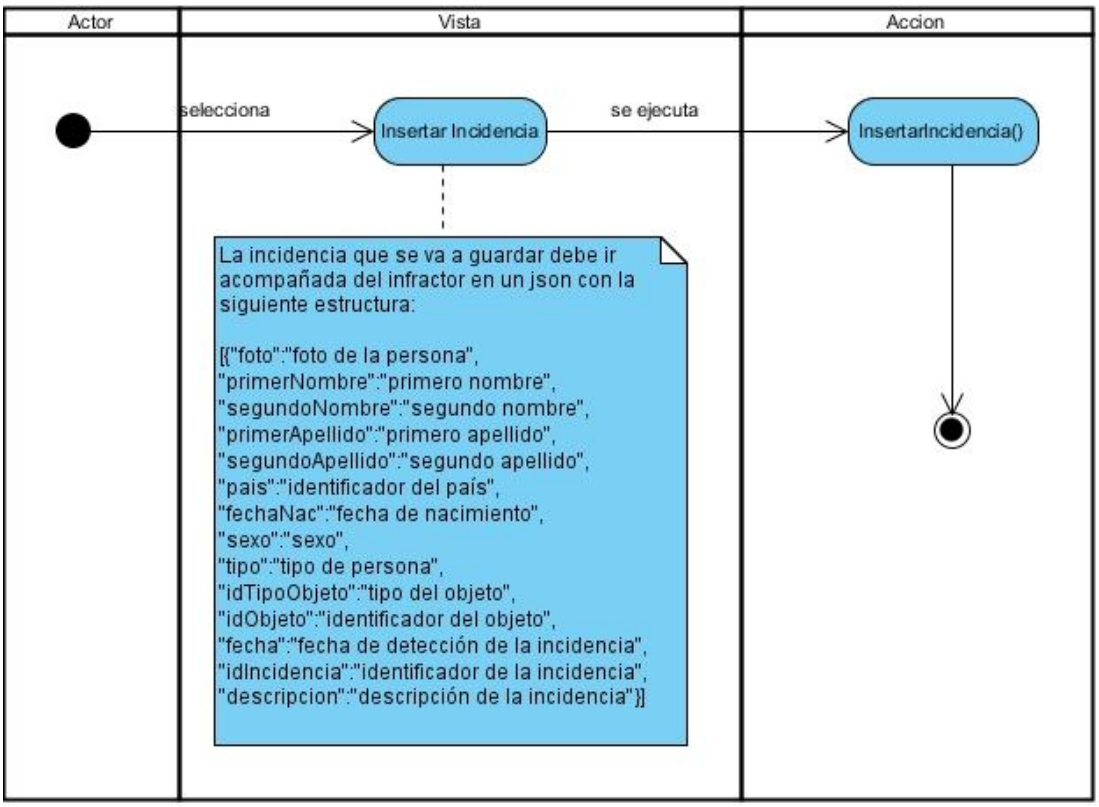


Diagrama de Secuencia orientado a las Actividades de la vista del RF Insertar Incidencia por Pantalla.

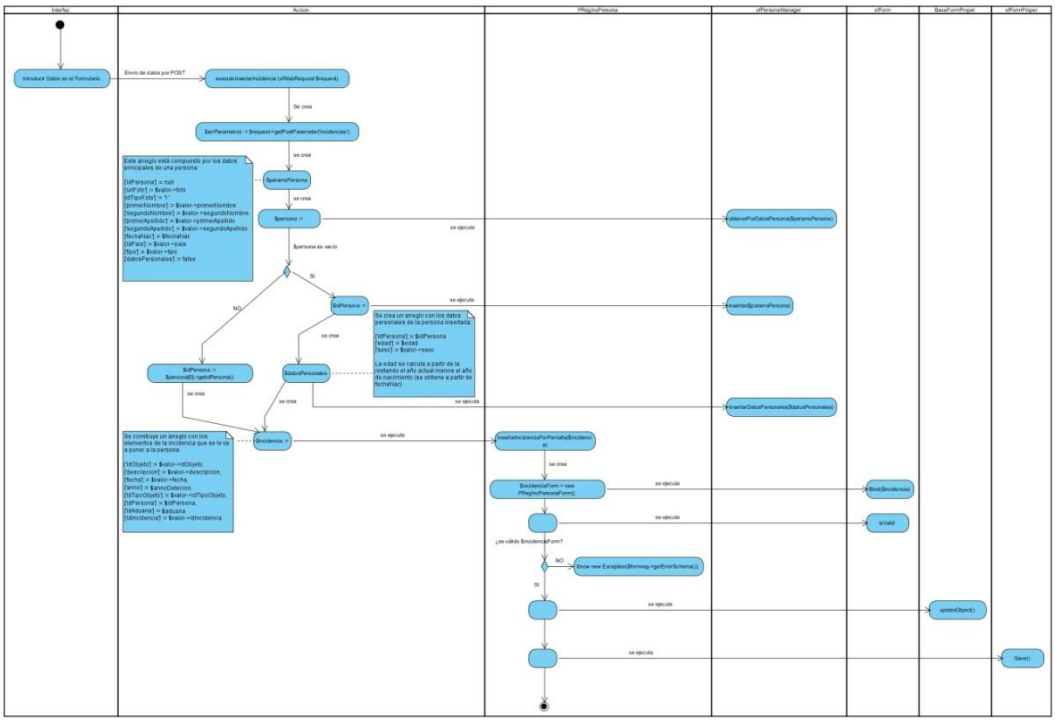


Diagrama de Secuencia orientado a las Actividades del Negocio del RF Insertar Incidencia por Pantalla.

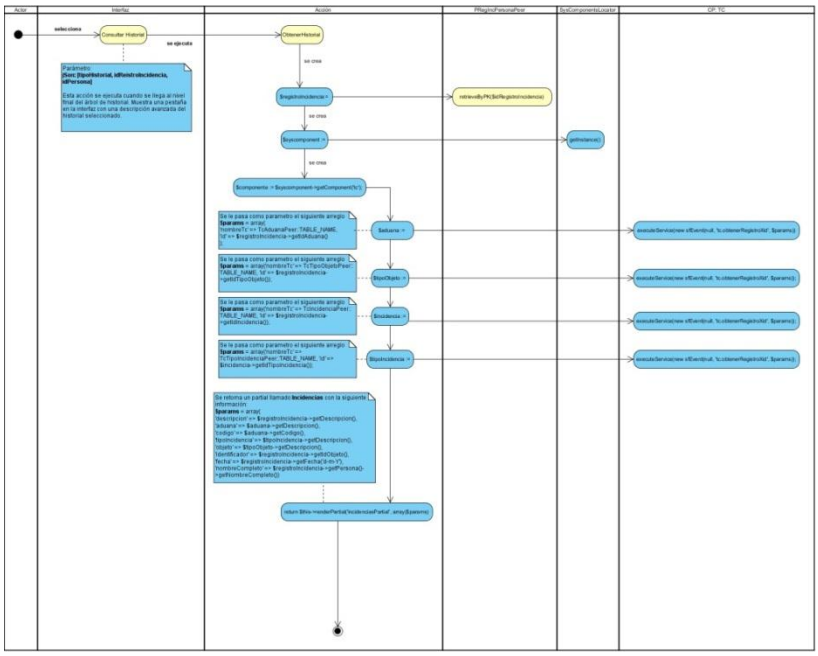


Diagrama de Secuencia orientado a las Actividades del Negocio del RF Consultar Historial de Incidencias de Persona Natural

Anexo #3: Certificado de aceptación

UCI Informáticas

CERTIFICADO DE ACEPTACIÓN

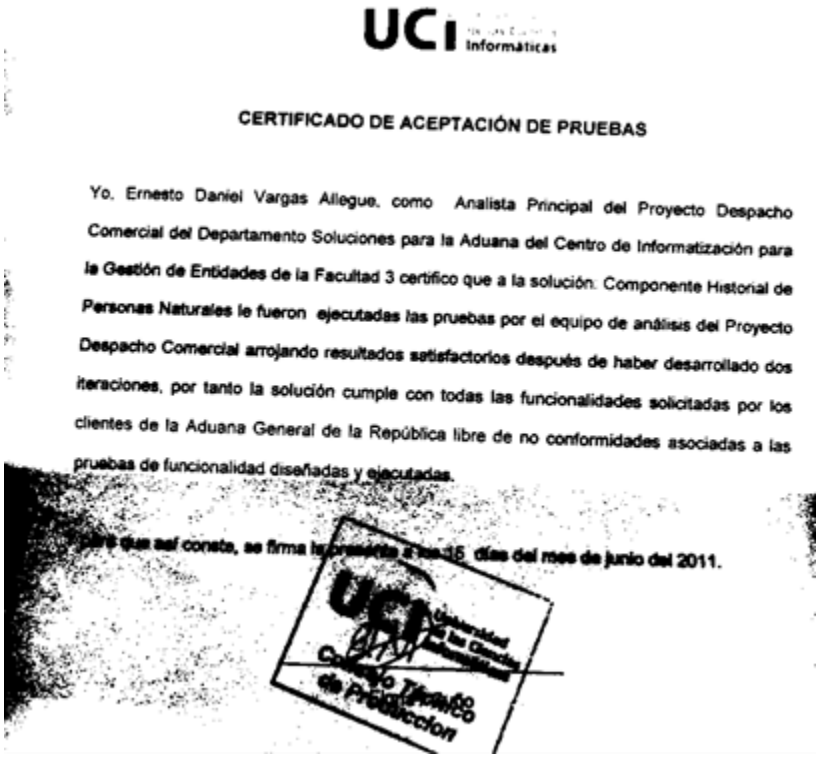
Yo. Alain Eduardo Rodríguez Arias, como Jefe del Departamento de Soluciones para la Aduana del Centro de Informatización para la Gestión de Entidades de la Facultad 3 certifico que la solución Componente Historial de Personas Naturales cumple con todas las funcionalidades solicitada por los clientes de la Aduana General de la República.

Y para que así conste, se firma la presente a los 8 días del mes de junio del 2011.

Firma / Cuño

Figura 21: Certificado de aceptación emitido por el Jefe de Departamento

Anexo #4: Certificado de aceptación de pruebas



UCI Informáticas

CERTIFICADO DE ACEPTACIÓN DE PRUEBAS

Yo. Ernesto Daniel Vargas Allegue, como Analista Principal del Proyecto Despacho Comercial del Departamento Soluciones para la Aduana del Centro de Informatización para la Gestión de Entidades de la Facultad 3 certifico que a la solución: Componente Historial de Personas Naturales le fueron ejecutadas las pruebas por el equipo de análisis del Proyecto Despacho Comercial arrojando resultados satisfactorios después de haber desarrollado dos iteraciones, por tanto la solución cumple con todas las funcionalidades solicitadas por los clientes de la Aduana General de la República libre de no conformidades asociadas a las pruebas de funcionalidad diseñadas y ejecutadas.

Para que así conste, se firma la presente a los 15 días del mes de junio del 2011.

UCI Universidad de las Ciencias Informáticas
Centro Técnico de Producción

Figura 22: Certificado de aceptación emitido por el grupo de calidad del proyecto Despacho Comercial