Universidad de las Ciencias Informáticas

Facultad 3

Título: Diseño e implementación de un Marco de Trabajo para el proceso de desarrollo de la Solución Tecnológica de SIGESAP.

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

Autores: Randy de Haz Martínez

Aldo Suárez Campos

Tutor: Ing. Lissuan Fadraga Artiles

Ciudad de la Habana, Junio 2011

”La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y, por regla general pueden ser expresadas en un lenguaje comprensible para todos”

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Randy de Haz Martínez

_____

Firma del Autor

Aldo Suárez Campos

_____

Firma del Autor

Ing. Lissuan Fadraga Artiles

_____

Firma del Tutor

Agradecimientos

A mis padres por haberme brindado todo su apoyo, su amor, y la mejor educación.

A toda mi familia que siempre me ayudó en todo lo que necesité durante los 5 años de estudio.

A esta maravillosa escuela obra de Fidel y la Revolución, que me ha formado como profesional y me ha permitido conocer personas de todo el país, junto a las que he pasado momentos maravillosos.

A todos los estudiantes y profesores del Proyecto Rap, sin los cuales no fuera posible el desarrollo de este trabajo.

A mi tutor Lissuan y compañero de tesis Randy, que día a día estuvieron ahí, para desarrollo de este trabajo de diploma.

Aldo.

A mi Mamá y mi Papá por ser guías, guardias, luz, faro, ejemplo, temple y cariño para mí.

A mi Hermana, por soportar mis maldades, y ser tan especial a la hora de escuchar.

A mi tío Juan Carlos, por ser padre, amigo, compañero, y un eterno joven.

A mi tío Eduardo, que aunque no esté presente, vive, y sé que está aquí conmigo.

A mi abuela María, por quererme y malcriarme tanto.

A mi tío Ernesto, por apoyarme cuando más lo he necesitado.

A mi otra familia de Camagüey, por haberme acogido como uno más.

A mis compañeros de la emisora HabanaRadio, que son personas geniales.

A mis amigos Yendri, Joao, Horlenys, Jorge Humberto, por ayudarme y ser apoyo.

A Dariela, por guiarme por el buen camino desde los inicios de la universidad.

A mi tutor Lissuan, por ser el goleador más grande de la UCI, y la Wikipedia del proyecto.

A mi compañero de tesis, Aldo, por compartir junto a mí los conocimientos adquiridos y llevarlos a la práctica.

Randy.

- *Dedicatoria*

A mis padres que son las personas más importantes en mi vida y la razón de mi existencia.

Aldo.

De manera especial a mis padres y familia en general, por hacerme sentir tan afortunado cada día de mi vida por tenerlos, por su apoyo, por su amor, por su paciencia, y por ser sobre todas las cosas, personas inigualables, especiales, fantásticas, unidas, personas con un corazón que no les cabe en el pecho.

Randy.

RESUMEN

Producto del convenio entre Cuba y la República Bolivariana de Venezuela, en la 10ma mixta, se firma el contrato Solución Tecnológica Integral para la Automatización y Modernización de la División de Antecedentes Penales de la República Bolivariana de Venezuela. Dicha solución comprende dos subsistemas Centro de Digitalización y Sistema de Gestión de Antecedentes Penales, los cuales debido al negocio que presentan poseen elementos en común, por lo que es necesario desarrollar un Marco de Trabajo que facilite el desarrollo de los dos subsistemas y sea capaz de gestionar el trabajo de los distintos módulos que se encuentran dentro de cada subsistema. Para dar solución a esta problemática se realizó este trabajo de diploma cuyo objetivo principal es implementar un Marco de Trabajo, que sea capaz de facilitar el desarrollo de los subsistemas de la solución para agilizar el trabajo entre ambos.

El proceso de desarrollo estuvo guiado por el Proceso Unificado de Desarrollo, y se basa en tecnologías libres y multiplataforma. Emplea Java como lenguaje de programación, NetBeans como IDE de Desarrollo, PostgreSQL como Sistema de Gestión de Bases de Datos y GlassFish Open Source como servidor de aplicaciones.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA. 4

1.1 SOLUCIONES EXISTENTES 4

1.1.1 MARCOS DE TRABAJOS UTILIZADOS EN EL MUNDO. 4

1.1.2 MARCOS DE TRABAJOS UTILIZADOS EN LA UCI 6

1.2 TENDENCIAS, TECNOLOGÍAS, HERRAMIENTAS Y ESTÁNDARES UTILIZADOS EN EL PROCESO 6

1.2.1 PATRONES DE ARQUITECTURA 7

1.2.2 SISTEMA OPERATIVO 10

1.2.3 LENGUAJE DE PROGRAMACIÓN 12

1.2.4 GESTOR DE BASE DE DATOS 14

1.2.5 LENGUAJE DE MODELADO 15

1.2.6 METODOLOGÍA DE DESARROLLO 16

1.2.7 HERRAMIENTA CASE 19

1.2.8 ENTORNO DE DESARROLLO 20

1.2.9 TECNOLOGÍAS 21

1.3 CONCLUSIONES 25

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA. 26

2.1 PROPUESTA DE SOLUCIÓN 26

2.2 ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE 27

2.2.1 REQUISITOS FUNCIONALES 27

2.2.2 REQUISITOS NO FUNCIONALES 28

2.3 PATRONES DE DISEÑO 30

2.3.1 PATRONES A UTILIZAR EN EL DESARROLLO 30

2.4 ESTÁNDARES DEL DISEÑO 31

2.5 DIAGRAMA DE CLASES DEL DISEÑO 33

2.5.1 DIAGRAMAS DE CLASES DEL FRAMEWORK PARA LA PARTE DEL CLIENTE. 33

2.5.2 DIAGRAMAS DE CLASES DEL FRAMEWORK PARA LA PARTE DEL SERVIDOR 38

2.5.3 DIAGRAMA DE CLASES DE LOS CASOS DE USO 39

2.6 Diagrama de clases de las acciones del sistema. 43
2.7 Diagrama de los Paquetes del Sistema 44
2.8 Descripción de Clases y Métodos del Paquete de Seguridad....45
2.9 Descripción de las Clases y Métodos del CU Buscar Persona....48
2.10 Modelo de Datos 53
2.11 Conclusiones 56
Capitulo 3: Implementación 57
3.1 Modelo de Implementación 57
3.1.1 Diagrama de Componentes 57
3.1.2 Diagrama de Integración del Marco de Trabajo con los Subsistemas 60
3.2 Tratamiento de excepciones....60
3.3 Seguridad 62
3.4 Estándares de Codificación 64
3.4.1 Nomenclatura para los paquetes, métodos y clases....64
3.4.2 Nomenclatura para los componentes de formularios 64
3.5 Conclusiones 65
Capitulo 4: Validación de la solución....66
4.1 Métricas de Software 66
4.1.1 Tamaño operacional de clase (TOC) 67
4.1.2 Relaciones entre clases (RC)....69
4.2 Pruebas de Software....72
4.3 Conclusiones 75
Conclusiones Generales....76
Recomendaciones 77
Bibliografía 78
Anexo1. Descripción de las clases....81
Anexo2. Imágenes 93

Introducción

El notable desarrollo alcanzado en las tecnologías de información y las comunicaciones para la prestación de servicios y el almacenamiento de grandes volúmenes de información ha propiciado que se utilicen en muchas esferas de la sociedad. La informática jurídica no ha sido la excepción, y es por ello que la Universidad de las Ciencias Informáticas (UCI) ha creado un Centro de Gobierno Electrónico (CEGEL). Este centro tiene el objetivo de crear productos, servicios y soluciones informáticas de alta calidad, vinculados en su mayoría a la Informática Jurídica.

Entre las soluciones informáticas que se desarrollan en CEGEL se encuentran el Sistema de Gestión para Seguimiento y Control de los Proyectos del convenio Integral de Cooperación Cuba-Venezuela (CCV), Sistema de Automatización y Modernización de Registros Principales y Notarías Públicas (SAREN) y la Solución Tecnológica Integral para la Automatización y Modernización de Antecedentes Penales de la República Bolivariana de Venezuela (RAP).

Este último surge producto del convenio Cuba-Venezuela durante la 10ma mixta, con el objetivo de desarrollar un software que permita digitalizar y gestionar la información existente, en la División de Antecedentes Penales de la República Bolivariana de Venezuela, así como también mejorar la prestación de los servicios de certificación de Antecedentes Penales y utilizar una tecnología multiplataforma. Para ello el proyecto cuenta con dos subsistemas, Centro de Digitalización y Sistema de Gestión de Antecedentes Penales, dentro de los cuales existen varios módulos, que tienen componentes comunes. En estos momentos no existe un mecanismo que agrupe el trabajo en conjunto de ambos subsistemas y por consiguiente el de los módulos, lo que atenta contra la reusabilidad que se exige, además de las consecuencias que esto trae al mantenimiento de los mismos. Por otro lado no existe un mecanismo que centralice los principales elementos que rigen los procesos fundamentales de la solución, como son la seguridad, el tratamiento de excepciones y la mensajería.

La situación anterior conlleva al siguiente **problema**: ¿Cómo centralizar los elementos comunes de ambos subsistemas, para facilitar el desarrollo de la Solución Tecnológica Integral para la Automatización y Modernización de la División de Antecedentes Penales?

El **objeto de estudio** al que se dirige la investigación es el proceso de desarrollo de software y como **objetivo general** se plantea realizar el diseño e implementación de un Marco de Trabajo que facilite el proceso de desarrollo de la Solución Tecnológica Integral para la Automatización y Modernización de la

División de Antecedentes Penales de la República Bolivariana de Venezuela. Este objetivo tiene como **campo de acción** Marcos de Trabajo para el desarrollo de aplicaciones multiplataforma.

Idea a defender:

Con el diseño e implementación de un Marco de Trabajo que centralice los elementos comunes de los subsistemas Centro de Digitalización y Sistema de Gestión de Antecedentes Penales se facilitará el proceso de desarrollo de la Solución Tecnológica Integral para la Automatización y Modernización de la División de Antecedentes Penales de la República Bolivariana de Venezuela.

Para lograr el objetivo general de la investigación se plantean los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Desarrollar el diseño e implementación de un Marco de Trabajo que facilite el proceso de desarrollo de la Solución Tecnológica Integral para la Automatización y Modernización de la División de Antecedentes Penales de la República Bolivariana de Venezuela.
- Validar la solución propuesta.

Para dar cumplimiento a los objetivos específicos planteados se definen las siguientes **Tareas de la investigación:**

- Análisis de las principales herramientas, metodologías y tecnologías para la solución del problema.
- Desarrollo de los artefactos de los flujos de trabajo Diseño e Implementación.
- Diseño del Modelo de Pruebas para el sistema y validación de los componentes implementados.

Métodos teóricos:

Analítico-Sintético: Ha hecho posible que a través del estudio de las necesidades del proyecto se arribe a la concesión de un Marco de Trabajo que brinde solución al trabajo paralelo entre ambos subsistemas.

Histórico-Lógico: Ha permitido realizar un estudio histórico sobre las aplicaciones que realizaban trabajos similares, para así poder tener un mayor conocimiento y saber aplicarlo en la solución del problema.

Estructura

La presente investigación está estructurada por 4 capítulos, en los cuales se tratan las cuestiones siguientes:

Capítulo 1: Fundamentación teórica. Tecnologías, metodologías y herramientas de desarrollo a utilizar.

Capítulo 2: Características del sistema. Descripción de la solución propuesta a la problemática planteada.

Capítulo 3: Implementación. Desarrollo de la solución propuesta del sistema.

Capítulo 4: Prueba. Validación de los resultados.

Capítulo 1: Fundamentación Teórica.

En el desarrollo de software, un Marco de Trabajo es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base 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 programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

En este capítulo se describe un estudio de Marcos de Trabajo en el ámbito nacional e internacional, así como también las tecnologías, herramientas y metodologías a utilizar para el desarrollo del software a partir de la arquitectura definida.

1.1 Soluciones Existentes

Hoy en día existe en el mercado una gran cantidad de framework de aplicaciones orientados a la gestión de proyectos, muchos de los cuales son ampliamente utilizados en organizaciones y empresas debido a las ventajas y comodidades que estos ofrecen. Una de las metas más importantes de este proceso es conseguir adaptar un paradigma de desarrollo a las necesidades y recursos del sistema, esto simplifica el desarrollo de una aplicación ya que son utilizados para resolver las tareas comunes, forzando al equipo de desarrollo a crear documentos y códigos más legibles y fáciles de mantener.

1.1.1 Marcos de trabajos utilizados en el mundo.

Spring

Es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. La primera versión fue escrita por Rod Jonhson, quien lo lanzó primero con la publicación de su libro Expert One-on-One Java EE Design and Development (Wrox Press, octubre 2002). El framework fue lanzado inicialmente bajo Apache 2.0 License en junio de 2003. El primer gran lanzamiento hito fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros hitos en septiembre de 2004 y marzo de 2005. Una de las metas de diseño de Spring Framework es su facilidad de integración con los estándares J2EE y

herramientas comerciales existentes. Esto quita en parte la necesidad de definir sus características en un documento de especificación elaborado por un comité oficial y que podría ser criticado. Después en 2005 Spring superó las tasas de adopción del año anterior como resultado de nuevos lanzamientos y más características fueron añadidas. En el mismo año los desarrolladores del proyecto abrieron su propia compañía para ofrecer soporte comercial y establecieron una alianza con BEA. Algunas de las ventajas de este framework son: (30).

- **POJO**: (Plain Old Java Object) revalora la simplicidad de las clases Java aportando manejo de transacciones de forma no intrusiva.
- **XML**: configuración basada en archivos XML.
- **Seguridad**: como un requerimiento no funcional implementado como un aspecto (AOP) a través del framework Acegi.
- **Remoting**: RMI simplificado, acceso y publicación de web services.
- **Testing**: provee un package de testing específico para componentes del framework e integrado con JUnit.

Asp.Net MVC

Es un framework metodológico que divide la implementación de una aplicación en tres roles: modelos, vistas y controladores. Uno de los beneficios de usar una metodología MVC es que ayuda a mantener una separación limpia entre modelos, vistas y controladores en la aplicación. Manteniendo una separación clara de conceptos hace que las pruebas de las aplicaciones sean mucho más fáciles, ya que los contratos entre diferentes componentes de la aplicación están mejor definidos y articulados. Algunas de las ventajas de este framework son:

- Es altamente extensible.
- Incluye un potente mapeador de URL que nos permite crear aplicaciones con URLs limpias.
- El framework MVC soporta los archivos .ASPX, ASCX y MASTER.

- Soporta todas las características de ASP.NET como ventanas de autenticación, autorización, mediante roles, mapeo de datos, administración del estado de la sesión y un sistema de configuración. (28).

1.1.2 Marcos de trabajos utilizados en la UCI

Dalas

La iniciativa en la creación de "Dalas" surge a raíz de un conjunto de problemas que se presentaron al utilizar soluciones informáticas de gran tamaño basadas en "Spring Framework". El framework ha evolucionado hasta conformar una base arquitectónica para las aplicaciones que utilizan "Spring Framework" actuando como integrador y manejador de los componentes de la aplicación.

Uno de los principales retos en las empresas constructoras de software es la reutilización de soluciones. En Dalas la reutilización constituye su principal razón de ser, para lo cual se han definido estándares sobre cada tipo de componente permitiendo mover las funcionalidades entre aplicaciones sin afectaciones.
Principales Funcionalidades:

- Gestión de módulos de negocio.
- Gestión de componentes reutilizables.
- Permite subdividir una aplicación para ser distribuida en varios servidores.
- Instanciación dinámica de una aplicación Web sobre "Spring Framework". (29).

1.2 Tendencias, tecnologías, herramientas y estándares utilizados en el proceso

En la actualidad existen muchas metodologías de desarrollo de software, lenguajes de programación, tecnologías de desarrollo, herramientas y estándares que pueden ser utilizados en soluciones informáticas. A continuación se hace un estudio en cada uno de los aspectos antes mencionados y se llega a una propuesta para la solución.

1.2.1 Patrones de Arquitectura

En el desarrollo de aplicaciones informáticas se utilizan los llamados patrones de arquitectura o estilos arquitectónicos que definen las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura fundamental de los sistemas. En una forma más específica, un estilo determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas.

Existen muchos patrones arquitectónicos, desde los que tratan problemas de manejo de Datos, presentación de los datos hasta el manejo de sesiones y concurrencia. Dentro de estos patrones es necesario hacer referencia a algunos que por sus características se deben tener en cuenta para la solución.

Sistema Modelo Cliente Servidor

El esquema cliente servidor es un modelo de computación en el que el procesamiento requerido para ejecutar una aplicación o conjunto de aplicaciones relacionadas se divide en dos o más procesos que cooperan entre sí. Usualmente la mayoría del trabajo pesado se hace en el proceso llamado servidor y el proceso cliente sólo se ocupa de la interacción con el usuario. Éste es un modelo que intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones. Comenzó a ser aceptado a finales de los años 80. Brinda como ventajas:

- La posibilidad de utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.
- Facilita la integración entre sistemas diferentes, permitiéndoles compartir información.
- El mantenimiento y el desarrollo de aplicaciones es rápido ya que se pueden emplear las herramientas existentes.
- No es siempre necesario transmitir información gráfica por la red, pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.

- La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones. (33).

Modelo Vista Controlador (MVC)

Sus características principales son que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Usado frecuentemente en aplicaciones web donde se utilizan diferentes interfaces de usuario. Las tres partes de este patrón se definen:

Modelo: Representa los datos del programa, manejándolos y controlando todas sus transformaciones. Lleva un registro de las vistas y controladores del sistema. Es independiente de cualquier representación de salida y/o comportamiento de entrada.
Vista: Maneja la presentación visual de los datos representados por el modelo. Muestra la información al usuario a través de una Interfaz de Usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
Controlador: Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, entre otros. Une la vista y el modelo. Interactúa con el modelo a través de una referencia al propio modelo.

La arquitectura Modelo Vista Controlador permite implementar cada uno de sus componentes por separado, donde tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. La conexión entre el modelo y sus vistas es dinámica, se produce en tiempo de ejecución, no en tiempo de compilación. Simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. (31).

Patrón Mapeo de Datos.

Por lo general el diseño de la base de Datos (BD) no guarda relación con el modelo de los datos, esto en parte viene dado por las diferencias conceptuales entre el modelo relacional y el modelo orientado a

objetos, surgiendo así uno de los tantos problemas comunes existentes en el desarrollo de aplicaciones informáticas. Para dar solución a esto surge el patrón de Mapeo de Datos.

El mapeo de los datos es una capa de software que separa los objetos en memoria de la base de datos. Su responsabilidad consiste en la transferencia de datos entre los dos y también para aislarlos unos de otros. Los objetos no tienen por qué saber que hay una base de datos para almacenar la información de ellos, no necesitan ningún código de interfaz SQL para hacerlo, y ciertamente ningún conocimiento del esquema de base de datos.

Arquitectura en Capas.

Es un estilo en el que se divide el sistema en capas. La principal ventaja de estilo consiste en facilitar la gestión de los cambios en un sistema. Solo basta con trabajar con el nivel que se necesite realizar el cambio sin tener que tocar el resto de los niveles. Por lo general se divide el sistema en tres capas fundamentales, presentación, lógica de negocio y acceso a datos. Este estilo facilita que cualquier cambio que se realice sobre alguna capa sea transparente respecto a las otras capas que se implementan en el sistema. La abstracción de los desarrolladores de los otros niveles es además otra de las ventajas, estos solo necesitan conocer la interfaz que conecta cada una de las capas. Este estilo arquitectónico es muy usado, ya que permite una gran escalabilidad en los sistemas que se desarrollan, se debe a que facilita la ampliación de los mismos. Dentro de este estilo se utiliza mucho la arquitectura en tres capas.

Capa de presentación: Esta capa es la que permite la interacción con los usuarios de la aplicación. Su función principal es gestionar los datos con el usuario.

Capa de Negocio: Aquí se establecen todas las reglas del negocio que deben ser cumplidas. Esta interactúa con la capa de presentación para recibir los datos provenientes del usuario, así como para enviar respuestas a dichas peticiones. Además se relaciona con la capa de Datos para gestionar la información que se almacena en los medios de almacenamiento.

Capa de Datos: En esta reside toda la información referente al negocio. Además es la encargada de la gestión de los mismos. (32).

1.2.2 Sistema Operativo

Un sistema operativo (SO) es el programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite la normal ejecución del resto de las operaciones. Uno de los propósitos del sistema operativo consiste en gestionar los recursos de localización y protección de acceso del hardware, hecho que alivia a los programadores de aplicaciones de tener que tratar con estos detalles.

Windows

Microsoft Windows es un sistema operativo gráfico para computadoras personales cuyo propietario es la empresa Microsoft. Las distintas versiones de Windows, que ofrecen un entorno gráfico amigable y sencillo, principalmente desde la versión Windows 95, han convertido a Windows en el sistema operativo más utilizado del mundo. Por eso la mayoría de las empresas fabricantes de hardware y software en el mundo prefieren desarrollar sus aplicaciones basadas en este sistema operativo.

Windows ha incorporado a través de sus diferentes versiones múltiples herramientas que se han convertido en estándares en la mayoría de los usuarios en el mundo. Así, Windows incorpora, entre otros software, herramientas como Internet Explorer y el Reproductor de Windows Media, los cuales se han convertido en el navegador de Internet y reproductor multimedia, respectivamente, más populares en el mundo. (34).

Ventajas

- Gran compatibilidad con Software.
- Gran compatibilidad con Hardware.
- Posee respaldo de las compañías de pago.
- Es fácil de aprender y de manejar.
- Mediante Windows Server 2003 se puede lograr una red informática segura y eficiente.
- El tiempo de desarrollo de aplicaciones y sistemas que sean ejecutables sobre servidores Windows es bastante rápido.
- Es el sistema operativo más comercial.
- Posee una infraestructura segura.

Desventajas

- Su precio es exagerado.
- Es un software propietario.
- Poca velocidad tras usarlo mucho tiempo.
- Pocas capacidades de personalización.

Linux

Como sistema operativo, Linux es muy eficiente y tiene un excelente diseño. Es multitarea, multiusuario, multiplataforma y multiprocesador; en las plataformas Intel corre en modo protegido; protege la memoria para que un programa no pueda hacer caer al resto del sistema; carga sólo las partes de un programa que se usan; comparte la memoria entre programas aumentando la velocidad y disminuyendo el uso de memoria; usa un sistema de memoria virtual por páginas; utiliza toda la memoria libre para cache; permite usar bibliotecas enlazadas tanto estática como dinámicamente; se distribuye con código fuente; usa hasta 64 consolas virtuales; tiene un sistema de archivos avanzado pero puede usar los de los otros sistemas; y soporta redes tanto en TCP/IP como en otros protocolos. (37).

Ventajas

- Es totalmente gratuito y aunque posee versiones con soporte técnico es aún más barato que comprar Windows.
- Alta velocidad y confiabilidad en las aplicaciones.
- La seguridad contra los Hackers y/o creadores de virus es alta ya que rara vez atacan a Software de Linux.
- Carga y realiza tareas con mayor eficiencia y velocidad que Windows.
- Existen infinidades de distribuciones de Linux, y con una constante actualización de las versiones.
- Posee protección de la memoria entre procesos, de manera que uno de ellos no pueda colgar el sistema.
- Posee un potente sistema de administración de memoria y de redes.

Desventajas

- El aprendizaje es lento
- Linux no cuenta con una empresa que lo respalde, por lo que no existe un verdadero soporte como el de otros sistemas operativos.
- La configuración de dispositivos de entrada y salida no es trivial.
- Poca disponibilidad de software.

Conclusiones

Basándose principalmente en la necesidad económica y en la velocidad, eficiencia y seguridad de los servidores en Linux, se decidió utilizar esta plataforma para el desarrollo del software.

1.2.3 Lenguaje de Programación

Los sistemas computacionales no se escriben en lenguaje natural (como se escribe o se habla comúnmente), por lo que se necesita un lenguaje que permita la comunicación entre la computadora y el desarrollador. Al lenguaje, en que se escriben los programas y que la computadora entiende se le llaman lenguajes de programación.

Desde el surgimiento de los lenguajes de programación han sido muchos los que se han desarrollado, cada uno con sus características específicas. Cada lenguaje tiene ventajas y desventajas que lo hacen más o menos potente ante una necesidad de software determinada, por lo que no se puede afirmar que exista un lenguaje que cubra todas las necesidades requeridas por un software.

C Sharp

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. Su sintaxis básica deriva de C y C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java aunque incluye mejoras derivadas de otros lenguajes. (35).

Ventajas

- Permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.
- Es simple y robusto ya que no tiene que preocuparse por apuntadores.
- Es familiar ya que la sintaxis de java es muy similar a la de lenguajes como C y C++ donde la mayoría de los programadores están acostumbrados a programar.
- Incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros aunque se puede violar estas restricciones.
- Potente librería gráficas.
- Aplicaciones Multi-hilo simplificadas.

Desventajas

- Fue diseñado para Windows.
- Su licencia es cara.
- Es complicado el manejo a bajo nivel

Java

Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. (36).

Ventajas

- Es independiente de la plataforma de desarrollo.
- Existen dentro de su librería clases gráficas como awt y swing, las cuales permiten crear objetos gráficos comunes altamente configurables y con una arquitectura independiente de la plataforma.
- Java permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.
- El manejo de las bases de datos es uniforme, es decir, transparente y simple.
- Es simple y robusto ya que el sistema de Java maneja la memoria.

- El sistema de Java es seguro ya que tiene ciertas políticas que evitan que se pueda codificar virus con este lenguaje y existen muchas restricciones que limitan lo que se puede hacer con los recursos críticos de una computadora.
- Es familiar ya que la sintaxis de java es muy similar a la de lenguajes como C y C++ donde la mayoría de los programadores están acostumbrados a programar.

Desventajas

- Los programas hechos en Java no tienden a ser muy rápidos.
- Para manejo a bajo nivel deben usarse métodos nativos, lo que limita la portabilidad.
- El diseño de interfaces gráficas con awt y swing no es simple.
- Algunas herramientas tienen un costo adicional.

Conclusiones

Debido a que ambos lenguajes de programación son potentes se decidió optar por la utilización de Java debido principalmente, a que es multiplataforma y su licencia es libre y la plataforma donde se desarrollará será Linux.

1.2.4 Gestor de Base de Datos

Los sistemas de gestión de bases de datos (en inglés database management system, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

El propósito general de los sistemas de gestión de bases de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

PostgreSQL

PostgreSQL es un Sistema Gestor de Base de Datos (SGBD) relacional de código abierto, muy poderoso, con una arquitectura probada. Puede ser ejecutado sobre la mayoría de los sistemas operativos que existen hoy en día. Posee protección de transacciones u operaciones de Atomicidad, Consistencia,

Aislamiento, Durabilidad (ACID, por sus siglas en inglés). Es un gestor de base de datos empresarial, que posee características sofisticadas como Control de Concurrencia Multi - Versión (MVCC, por sus siglas en inglés), replicación asíncrona, transacciones anidadas, realización de respaldo de datos en línea, optimizador o planificador de consultas, soporta internacionalización. Es altamente escalable en cuanto a la cantidad de información que puede manejar y al número de usuarios concurrentes que puede alojar.
El tamaño máximo para las bases de datos PostgreSQL es ilimitado. Es totalmente compatible con el estándar ANSI-SQL 92/99. Permite realizar relaciones de herencia entre tablas, definir sistemas de reglas y eventos. PostgreSQL ejecuta procedimientos de almacenados en más de una docena de lenguajes de programación tales como Java, Perl, Python, Ruby, Tcl, C/C++, y su propio lenguaje PL/pgSQL. De esto se deriva que existan muchas librerías para realizar la conexión a PostgreSQL desde varios lenguajes, compilados o interpretados, entre ellos ODBC, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme, para nombrar algunos. Desde el año 1999 hasta la fecha ha ganado un gran número de premios que ratifican a este SGBD como el más completo dentro del mundo del código abierto por lo que fue seleccionado como gestor de base de datos a utilizar.(5)

1.2.5 Lenguaje de Modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar (parte de) un diseño de software orientado a objetos.

Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo. Esto puede suponer también que las interacciones entre partes del programa den lugar a sorpresas cuando el modelo ha sido convertido en un software funcional. (10).

UML

Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, Unified Modeling Language), es utilizado para especificar, visualizar y documentar los artefactos de un sistema de desarrollo de software Orientado a Objetos. Unifica principalmente los métodos de Booch, Rumbaught (OMT) y Jacobson. Es el

lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía se encuentra en proceso de estandarización por el OMG (Object Management Group). Ya que permite modelar estructuras complejas, se adecua a las necesidades de conectividad actual y futura, es concurrente, distribuido y genera código a partir de los modelos y a la inversa. Además cuenta con varios tipos de diagramas los cuales muestran diferentes aspectos de las entidades representadas.

UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. (6).

1.2.6 Metodología de Desarrollo

Una metodología de desarrollo de software es un proceso en el que se define “quién” está haciendo “qué”, “cómo” y “cuándo”. Existen dos tipos de metodologías, ágiles y pesadas. Las metodologías ágiles enfatizan las comunicaciones cara a cara en vez de la documentación, y también, en que el software funcional es la primera medida del progreso. Las metodologías pesadas en cambio se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano. A lo largo del tiempo, una gran cantidad de métodos han sido desarrollados diferenciándose por su fortaleza y debilidad, pero en general consisten en:

- Una filosofía de desarrollo de programas de computación con el enfoque del proceso de desarrollo de software.
- Herramientas, modelos y métodos para asistir al proceso de desarrollo de software.

Con ánimos de buscar una metodología que se ajuste al medio en que se desarrollará el proyecto se analizaron algunas de las más usadas en el mundo. Dentro de estas están Programación extrema (XP, por sus siglas en inglés), Proceso Unificado de Rational (RUP, por sus siglas en inglés) y Microsoft Solution Framework (MSF, por sus siglas en inglés).

XP

XP (Extreme Programing) nace como nueva disciplina de desarrollo de software, y ha conseguido el respaldo de gran parte de la industria del software y el rechazo de otra parte. La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica. Esta es una metodología para un ágil desarrollo de software, con una programación basada en los deseos del cliente. El equipo lo conforman los jefes de proyecto, desarrolladores y el cliente, y se rige por valores y principios. (7)(9).

Ventajas

- Programación organizada.
- Menor taza de errores.
- Satisfacción del programador.

Desventajas

- Es recomendable emplearlo solo en proyectos pequeños y con un corto plazo.
- Altas comisiones en caso de fallar.
- Es difícil realizar mejoras o captar personal nuevo cuando no se realiza la documentación.
- Es recomendable solo para programadores avanzados.

MICROSOFT SOLUTION FRAMEWORK (MSF)

MSF es un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. Más que una metodología rígida de administración de proyectos, MSF es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información. Lo que permite a la empresa determinar los criterios de planeación más eficientes aplicando estos métodos en los diversos problemas que se presenten.

Ventajas

- MSF ayuda a implantar soluciones en base a la tecnología.
- Tiene facilidad de soporte y mantenimiento.
- Efectúa trabajo en equipo y la colaboración.
- Se puede utilizar para proyectos de cualquier magnitud.
- Crea una disciplina de análisis de riesgos que ayuda y evoluciona con el proyecto.
- Cuenta con plantillas que ayudan para el proceso de documentación.

Desventajas

- Este modelo utiliza demasiada documentación en sus fases.
- El análisis de riesgo suele llevar mucho tiempo frenando el avance del proyecto.
- Al ser un modelo de Microsoft implica que se tiene que utilizar herramientas solo de Microsoft.

RUP

El Proceso Racional Unificado (RUP, por sus siglas en inglés Rational Unified Process) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado. Proporciona una guía en el orden de las actividades de un equipo, dirige las tareas individuales de los desarrolladores, especifica qué productos deberían ser desarrollados y ofrece criterios para monitorear y medir los productos y actividades del proyecto. (7)(9).

Ventajas

- Posee una forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Evaluación en cada fase que permite cambios de objetivos.
- Seguimiento detallado en cada una de las fases.
- Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.

Desventajas

- La evaluación de riesgos es compleja.
- Excesiva flexibilidad para algunos proyectos.
- Genera muchos artefactos lo que significa un incremento de tiempo y de costos.

Conclusiones

Producto de que el proyecto es grande, no todos los desarrolladores son avanzados y podrían estar sujetos a cambios, de que se cuenta además con un largo plazo con el cliente y RUP ofrece una forma disciplinada de asignar tareas y responsabilidades se decidió utilizar esta metodología de desarrollo.

1.2.7 Herramienta Case

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. También presenta licencia gratuita y comercial.

Teniendo en cuenta las facilidades de esta herramienta para el trabajo colaborativo, además de la integración para el trabajo con modelos relacionales de Bases de Datos, modelado de procesos de negocio a través de BPMN así como el modelado de todos los artefactos propuestos por la metodología a través de UML, en una sola herramienta, se utilizará como herramienta CASE ya que cumple con todas las necesidades para desarrollar la solución. (4).

1.2.8 Entorno de Desarrollo

Un entorno de desarrollo integrado es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación y es posible que un mismo IDE pueda funcionar con varios lenguajes de programación.

Eclipse

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE). (2).

Ventajas

- Es un IDE multilenguaje y adaptable.
- Es potente, fiable y versátil.
- Es software libre y gratuito.
- Es automatizado y rápido.
- Posee una compilación incremental automática del código para reducir los tiempos de depuración y pruebas.

Desventajas

- Eclipse no incluye un diseñador visual de interfaces gráficas.
- Eclipse no incorpora plug-ins para comunicarse con servidores de aplicaciones J2EE.
- Falta de soporte a los Enterprise JavaBeans, a las Java Server Pages y a los servlets es total. No se incluyen asistentes o plug-ins para la edición o depuración de estos.

Netbeans

NetBeans es un entorno de desarrollo fundado por la Sun MicroSystems en junio del 2000. NetBeans IDE es un producto libre y gratuito sin restricciones de uso y cuenta con una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles) y es un sistema de proyectos basado en Ant, control de versiones y refactoring. (1).

Ventajas

- Es un IDE multilenguaje y adaptable.
- Es software libre y gratuito.
- Intuitivo y fácil de utilizar.
- Es rápido para aplicaciones de escritorio.
- Se puede desarrollar todo tipo de aplicaciones.
- Es poderoso y extensible.
- No hay que estar buscando plug-ins por todas partes, ya que por lo regular todo se encuentra en la misma caja.
- Fácil integración con el servidor de aplicaciones Glassfish.

Desventajas

- Consume gran cantidad de recursos.
- No posee un buen soporte de refactorización.

Conclusiones

Debido a que ambos IDE son potentes y completos, el criterio fundamental sobre el que se basó la selección de Netbeans como entorno de desarrollo fue la experiencia de los desarrolladores con el mismo.

1.2.9 Tecnologías

Las tecnologías existentes para el desarrollo de software no son perfectas y están concebidas para las necesidades de sistemas específicos que se pretendan construir. Estas a su vez están representadas por

lenguajes de programación, aunque la tendencia de los últimos años demuestra que estas proporcionan gran interoperabilidad con otros lenguajes de programación y tecnologías en general.

Enterprise Java Beans

Para Java existen tres categorías que encapsulan las tecnologías existentes. Estas categorías son las siguientes:

- Java Stadard Edition (JSE): Dentro de esta categoría se aglomeran todas las tecnologías existentes en Java para sistemas comunes o estándares.
- Java Enterprise Edition (JEE): Esta categoría agrupa todas las tecnologías que proporciona Java para la implementación de aplicaciones empresariales.
- Java Movile Edition (JME): Esta tecnología contiene las tecnologías que proporciona Java para aplicaciones móviles.

El diseño de la solución estará centrado en el uso de la categoría JEE, especialmente en la tecnología Enterprise Java Beans (EJB).

Enterprise Java Beans (EJB) es la tecnología del lado del servidor para la arquitectura de componentes de la Plataforma Java, (Java EE). La tecnología EJB permite el desarrollo rápido y simplificado de aplicaciones distribuidas, transaccionales, seguras y portátiles basadas en tecnología Java. EJB 3.0.El contenedor EJB proporciona dos formas de interacción entre los componentes, una es a través de Interfaces Remotas a métodos (RMI, por sus siglas en inglés) y la otra es a través de Servicios Web (WS, por sus siglas en inglés).

El contenedor EJB implementa un conjunto de servicios que serán de gran ayuda para solución, evitando así la implementación de estos. A continuación se muestran alguno de los servicios implementados por el contenedor EJB:

- Manejo de transacciones: apertura y cierre de transacciones asociadas a las llamadas a los métodos del bean.
- Seguridad: comprobación de permisos de acceso a los métodos del bean.
- Concurrencia: llamada simultánea a un mismo bean desde múltiples clientes.

- Servicios de red: comunicación entre el cliente y el bean en máquinas distintas.
- Gestión de recursos: gestión automática de múltiples recursos, como colas de mensajes, bases de datos o fuentes de datos en aplicaciones heredadas que no han sido traducidas a nuevos lenguajes/entornos y siguen usándose en la empresa.
- Persistencia: sincronización entre los datos del bean y tablas de una base de datos.
- Gestión de mensajes: manejo de Java MessageService (JMS).
- Escalabilidad: Posibilidad de constituir clúster de servidores de aplicaciones con múltiples hosts para poder dar respuesta a aumentos repentinos de carga de la aplicación con sólo añadir hosts adicionales.
- Adaptación en tiempo de despliegue: posibilidad de modificación de todas estas características en el momento del despliegue del bean. (21) (24).

API

API o interfaz de programación de aplicaciones (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.

Java Message Service (JMS)

Conocida por sus siglas JMS, es la solución creada por Sun Microsystems para el uso de colas de mensajes. Este es un estándar de mensajería que permite a los componentes de aplicaciones, basados en la plataforma JEE, crear, enviar, recibir y leer mensajes. También hace posible la comunicación de manera síncrona y asíncrona en ambos casos. Existen dos modelos de la API JMS, los cuales son:

Modelo Punto a Punto: Este modelo cuenta con solo dos clientes, uno que envía el mensaje y otro que lo recibe. Este modelo asegura la llegada del mensaje ya que si el receptor no está disponible para aceptar el mensaje o atenderlo, de cualquier forma se le envía el mensaje y este se encola en una pila del tipo FIFO para luego ser recibido según haya entrado.

Modelo Publicador/Suscriptor: Este modelo cuenta con varios clientes, unos que publican temas(tópicos) o eventos, y los que ven estos tópicos, a diferencia del modelo punto a punto este modelo tiende a tener más de un consumidor. (26).

Java Persistence (JPA)

Es la API de persistencia desarrollada para la plataforma Java EE. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos y permitir usar objetos regulares. El principal factor de decisión de esta API es la integración que tiene con la tecnología que se utilizará para el desarrollo de la solución. (18).

GlassFish 2.1

GlassFish es un servidor de aplicaciones de software libre, que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Es gratuito y de código libre, se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL.
GlassFish está basado en el código fuente donado por Sun y Oracle Corporation, tiene como base al servidor Sun Java System Application Server de Oracle Corporation, un derivado de Apache Tomcat, y que usa un componente adicional llamado Grizzly que usa Java NIO para escalabilidad y velocidad.(3)(25)

Swing

Es un conjunto de clases de Java que simplifica la construcción de aplicaciones de escritorio. Permite tener un sistema de ventanas y componentes gráficos, independiente del sistema operativo y la librería de dibujo que se tengan disponibles en la máquina clientes. Swing es un extenso conjunto de componentes que van desde los más simples, como etiquetas, hasta los más complejos, como tablas, árboles, y documentos de texto con estilo. Casi todos los componentes Swing descienden de un mismo padre llamado JComponent que desciende de la clase de AWTContainer. La característica más notable de los componentes Swing es que están escritos al 100% en Java y no dependen de componentes nativos.

1.3 Conclusiones

La fundamentación teórica es esencial en el desarrollo de toda investigación, ya que permite un mejor entendimiento de la solución y una mejor elección de las herramientas, metodologías y tecnologías a utilizar. Producto de esta investigación fue seleccionado LINUX como sistema operativo, JAVA como lenguaje de programación, NetBeans como IDE de desarrollo, UML como lenguaje de modelado, Visual Paradigm como herramienta CASE, RUP como metodología de desarrollo y Postgres como Gestor de Base de Datos. Durante el desarrollo de este capítulo se realizó un análisis de varios marcos de trabajos existentes y se fundamentó todo lo necesario para darle una solución eficaz al problema a resolver.

Capítulo 2: Características del Sistema.

En este capítulo se abordan las características de la solución propuesta para el sistema. Se describen los requisitos funcionales y no funcionales, así como los estándares del diseño. Además se muestran el modelo de datos y el diagrama de clases del diseño.

2.1 Propuesta de Solución

El Marco de trabajo tendrá una arquitectura cliente servidor, distribuida en capas y orientada al desarrollo de componentes.

Cliente

- **Gestor Marco de Trabajo**: Paquete que se encargará de estandarizar el formato de las distintas interfaces de usuario, manteniendo el color y el tamaño. Además gestiona el wizard, el menú, y el flujo de acciones de los botones anterior, siguiente, terminar y cancelar.
- **Componentes**: Paquete que contendrá los componentes que utilizará el Marco de trabajo como son el wizard, el menú y la barra de herramientas.
- **Internacionalización i18N**: Paquete que permitirá diseñar el software de manera tal que pueda adaptarse a diferentes idiomas y regiones sin la necesidad de realizar cambios de ingeniería ni en el código**.**
- **Impresión**: Paquete que contendrá las clases que permitirán la impresión de documento.
- **Mensajes**: Paquete que contendrá las clases con los mensajes personalizados que se le mostrarán al usuario final.
- **Presentación**: Contendrá dos paquete fundamentales, el paquete de acciones y el de formularios. En el primero se implementarán las acciones de todos los casos de uso, y en el segundo se diseñarán las interfaces de usuario aplicando las pautas de diseño.
- **Seguridad**: Paquete encargado de la seguridad del sistema. Aquí se implementarán las clases encargadas de la autenticación, la encriptación y la autorización.

Servidor

- **Traza**: Paquete que permitirá guardar las trazas en la base de datos de las acciones que realicen los usuarios que deban ser archivadas.
- **Seguridad**: Controlará el acceso a los métodos y clases por roles y grupos de usuarios.
- **Acceso Datos**: Paquete que implementará la lógica de negocio de acceso a datos correspondiente a los elementos del Marco de Trabajo.
- **Administración y configuración**: Módulo que permitirá gestionar la administración y configuración del Marco de Trabajo.

2.2 Especificación de Requisitos de Software

Durante la primera etapa de desarrollo de Software se extraen los requisitos de software. Estos son de gran importancia ya que permiten elaborar un producto de software de mayor calidad. Los requisitos de software son definidos como condiciones o capacidades que deben estar presentes en un sistema o componentes de éste, para satisfacer un contrato, estándar, especificación u otro documento formal, según el Glosario de la IEEE. Los requisitos pueden dividirse en requisitos funcionales y requisitos no funcionales.

2.2.1 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con que propiedades o cualidades se relacionen y muestran como los casos de uso serán llevados a la práctica. A partir de los procesos de negocio estudiados y las actividades a automatizar identificadas, se definieron los siguientes requisitos funcionales:

RF1 Gestionar las excepciones que lanza el sistema.
RF2 Garantizar la autenticación al sistema.
RF3 Enviar mensajes.
RF4 Recibir mensajes.

RF5 Imprimir documentos.
RF6 Gestionar el formato de las interfaces.
RF7 Crear menú a partir de roles de usuario autenticado.
RF8 Obtener texto de mensajes.
RF9 Obtener texto de etiquetas.
RF10 Terminar flujo de trabajo.
RF11 Cancelar flujo.
RF12 Avanzar un paso en el flujo.
RF13 Retroceder un paso en el flujo.
RF14 Validar el acceso a los métodos y clases por roles.
RF15 Guardar usuario autenticado.
RF16 Acceder a usuario autenticado.
RF17 Adicionar trazas.

2.2.2 Requisitos no Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades se deben concebir para obtener un sistema atractivo, usable, rápido y confiable. Por lo que ayudarán a definir si un producto es bien aceptado o poco aceptado. A continuación se especifican los requisitos no funcionales que se tendrán en cuenta para elaborar el sistema:

Usabilidad

- Existirán servidores locales con capacidad necesaria para el procesamiento de las solicitudes del conjunto de aplicaciones de las diferentes oficinas los cuales tendrán conexión con los servidores centrales para mantener la actualización de los datos en ambos sentidos.

Soporte

- El sistema debe ser multiplataforma.

Restricciones de Diseño

- El sistema se desarrollará en plataforma Linux.

- El lenguaje de programación es Java.
- La herramienta IDE de desarrollo utilizada será Netbeans.
- La herramienta case utilizada es el Visual Paradigm.
- La herramienta gestor de base de datos es el PostgreSQL.

Requisitos de Apariencia o Interfaz Externa

- El sistema tiene que ofrecer una interfaz amigable y fácil de operar.
- La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización.
- La entrada de datos incorrecta será detectada claramente e informada al usuario.
- Todos los textos y mensajes en pantalla aparecerán en idioma español.

Software

- **Cliente**

– Sistema Operativo: Ubuntu 10.10 o Windows XP.

– Máquina Virtual de Java: 1.6.

– Visor de Documentos PDF:

– Windows: Acrobat Reader.

– Ubuntu: Evince

– Herramientas Ofimáticas:

– OpenOffice 3.0.

- **Servidor**

– Máquina Virtual de Java: 1.6.

– Servidor de Aplicación: Glassfish 2.1.

– Gestor de Bases de Datos: PostgreSQL 8.4.

2.3 Patrones de Diseño

Los patrones de Diseño son una forma estandarizada de resolver problemas específicos de programación para mejorar la calidad y diseño del sistema. Ayudan a reutilizar las piezas de software, a un sistema en particular o a varios sistemas. Están estrechamente ligados a la Programación Orientada a Objetos (POO), ya que se valen de varios conceptos de la misma para definir las estrategias para su implementación. Normalmente con la POO se piensa que se tiene todo, pero como no se conoce la aplicación de los patrones de Diseño entonces en ocasiones se reinventa la rueda. Estos patrones se categorizan en tres tipos: Patrones de Creación, Patrones de Estructura y Patrones de Comportamiento.

2.3.1 Patrones a utilizar en el desarrollo

Los siguientes patrones resultarán de gran importancia en el desarrollo del marco de trabajo ya que permiten minimizar el tiempo y el costo del proyecto, además permiten la reutilización de código y reducir el impacto de los cambios entre otras ventajas.

- **Singleton:** Permitirá la creación de una única instancia de las clases que sean necesarias, como los gestores del negocio, y las factorías.
- **Facade:** Permitirá la comunicación entre las capas, con un bajo acoplamiento entre estas.
- **Bajo Acoplamiento:** Permitirá estimular la asignación de responsabilidades de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. El Bajo Acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables.
- **Experto:** Se utilizará para que cada objeto realice la funcionalidad de acuerdo a la información que domina, la cuestión a la hora de diseñar es asignar responsabilidades a la clase que mayor información posee para cumplir con dicha tarea.
- **Creador:** Permitirá encontrar un creador al que se debe conectar con el objeto producido en cualquier evento. Brinda soporte a un bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

- **Alta cohesión**: Permitirá brindar una alta cohesión funcional.
- **Composite:** Permitirá construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol, esto se aplicará en la interacción de varias acciones para la formación de una.
- **Envoltorio**: Se utilizará para añadir dinámicamente funcionalidad a un Objeto. Esto nos permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera, como es el caso del menú.
- **Proxy:** Se utilizará como intermediario para acceder a un objeto, permitiendo controlar el acceso a él.
- **Template Method:** Permitirá definir una estructura algorítmica en la súper clase, delegando la implementación a las subclases. Es decir, define una serie de pasos, en donde los pasos serán redefinidos en las subclases.

2.4 Estándares del diseño

Los estándares de diseño definen un conjunto de reglas para diseñar las interfaces del sistema, los cuales permiten obtener un diseño de alta calidad y que cumpla con las buenas prácticas establecidas en la Ingeniería de Software. A continuación se describen los estándares de diseño que se utilizarán para el desarrollo del sistema.

- La separación de los componentes respecto a los bordes del área de trabajo (margen izquierdo, derecho, superior e inferior) debe ser de 10 píxeles como mínimo.
- En un formulario, los datos asociados a una misma cosa se agruparán utilizando paneles con título. El título del panel se escribe en formato de oración, en negrita y sin dos puntos al final.
- Cuando el formulario tiene varios componentes. El nombre de las etiquetas se escribe encima de los componentes. El texto en las etiquetas se escribe en formato de oración, sin utilizar negrita y terminado en dos puntos. Los componentes irán debajo, a una distancia de 8 píxeles, cada componente de entrada de datos debe tener una altura de 23 píxeles. La separación horizontal

entre componentes debe ser de 20 píxeles y vertical de 15 píxeles. Todo alineado a la izquierda, y en caso de que aparezcan unos debajo de los otros se debe tratar de que todos los componentes tengan un mismo tamaño (el del mayor componente).

- En caso de que el formulario tenga pocos componentes, se pondrá primero la etiqueta y al lado el componente, tanto los componentes como las etiquetas con las mismas pautas anteriores. Además se alinean las etiquetas a la izquierda y los componentes también a la izquierda.
- Cuando la etiqueta no es para una entrada de datos, sino para mostrar los mismos, entonces irá en negrita.
- El texto de las etiquetas, títulos de columnas de tablas, botones, entre otros. debe ser siempre en formato de oración.
- Los combos deben mostrar como texto inicial –Seleccione--. Los ítems para seleccionar deben escribirse en formato de oración y deben aparecer en orden alfabético, a no ser que representen un flujo, en ese caso aparecerían de acuerdo al orden correspondiente. Cuando haya una opción Otros, estará ubicada al final.
- Cuando haya que introducir una fecha debe ponerse un componente para seleccionar la misma, evitar las entradas manuales del usuario.
- Los Botones siempre estarán situados en la parte inferior derecha de los formularios, con una altura de 25 píxeles y el ancho depende del Caption, siempre dejando un espacio de 8 píxeles delante y detrás de la palabra o el ícono. Sólo en el caso de los mensajes de confirmación, avisos y error los botones irán en el centro. El texto que contienen debe estar en formato de oración y sin utilizar negrita. Cuando aparezcan varios botones uno al lado del otro, de ser posible todos deben tener el mismo ancho y la separación entre ellos debe ser de 10 píxeles. El ancho mínimo de los botones debe ser de 75 píxeles.
- Los botones no llevan íconos, el Anterior y Siguiente llevan los caracteres <,> respectivamente.
- Los botones de SI y NO en los mensajes de confirmación deben llevar íconos y tendrán un ancho de 50 píxeles.

2.5 Diagrama de Clases del Diseño

El diagrama de clases del sistema es una descripción del sistema que se basa en mostrar las clases que componen al mismo. Es una de las descripciones más detalladas de un software porque en el mismo se muestran las clases, con sus funcionalidades y sus relaciones.

2.5.1 Diagramas de Clases del Framework para la parte del Cliente.

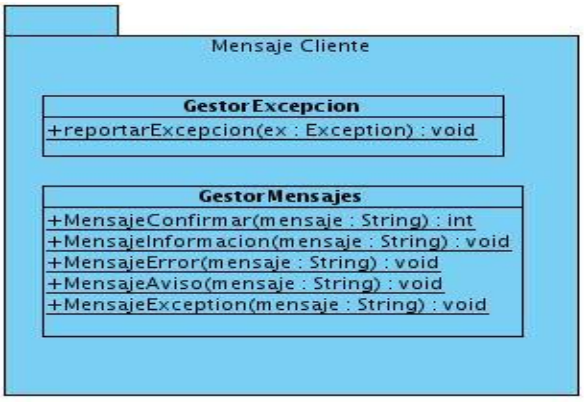


Diagrama de clases del componente Mensaje.

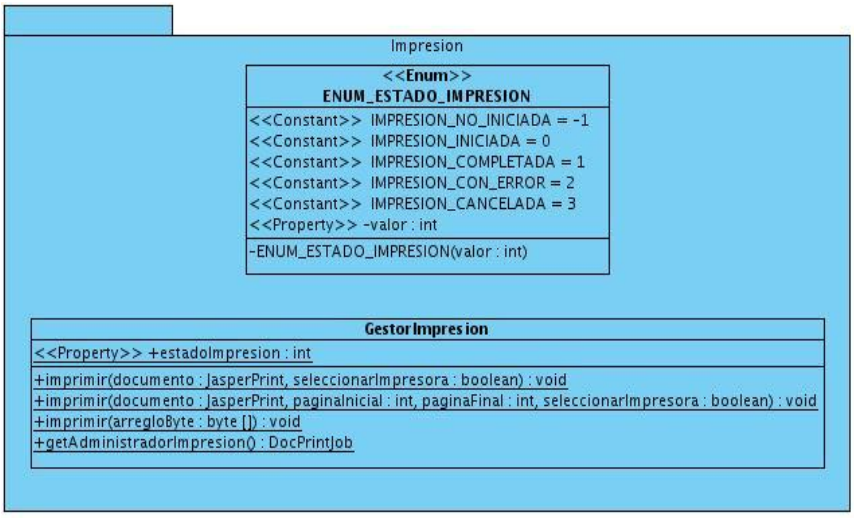


Diagrama de clases del componente de Impresión.

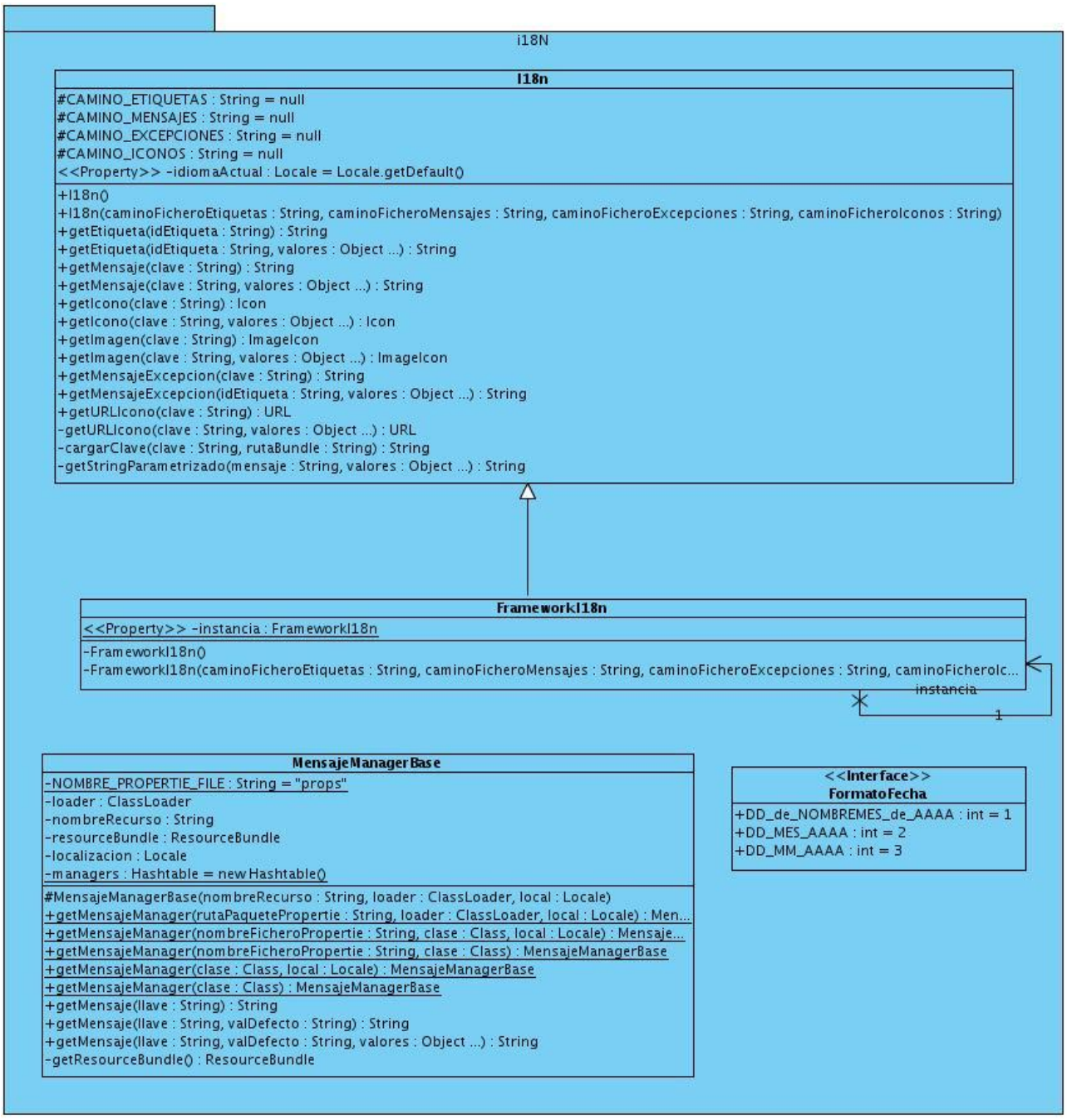


Diagrama de clases del componente de internacionalización i18N.

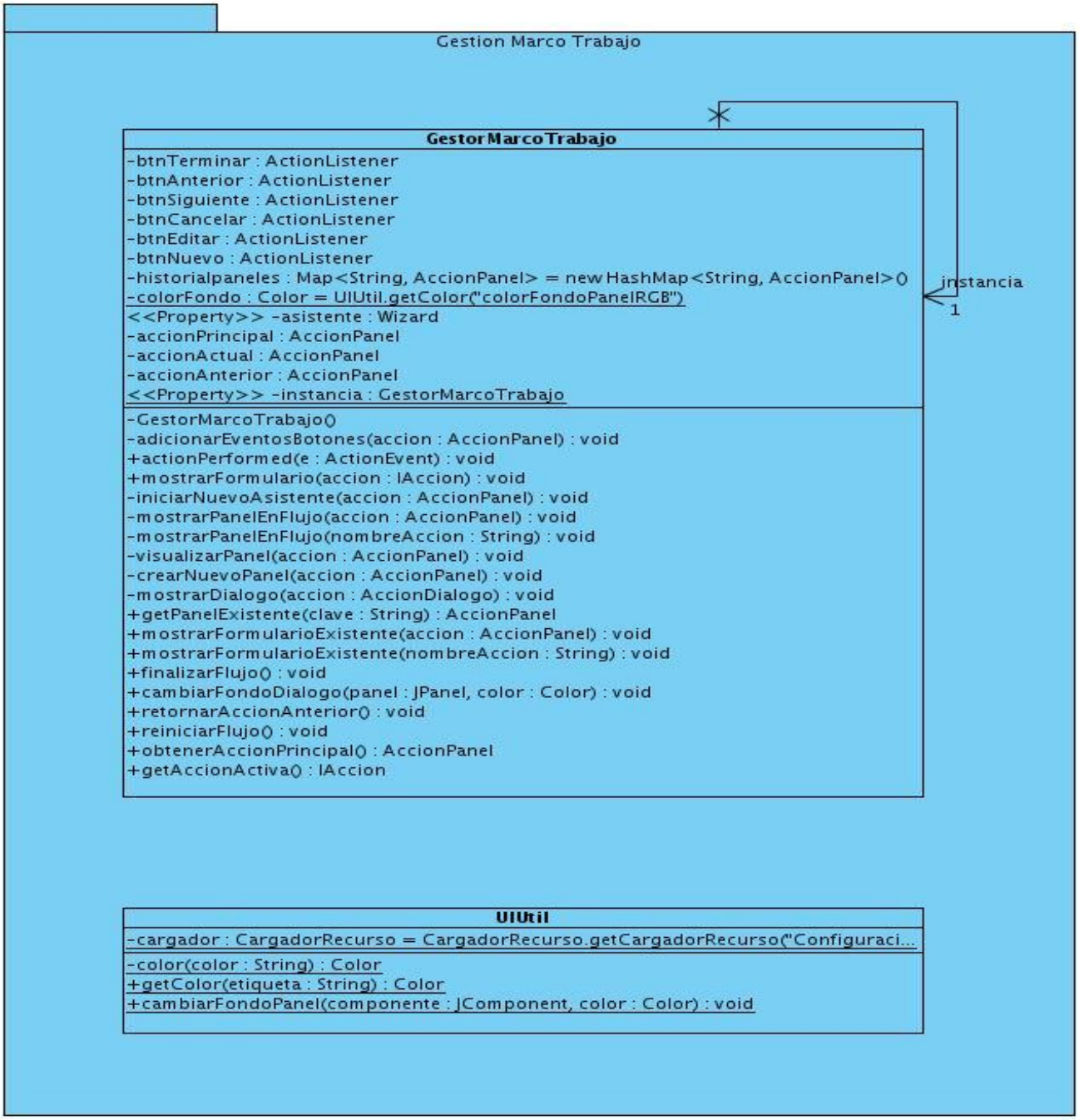


Diagrama de clases del componente Gestión Marco de Trabajo.

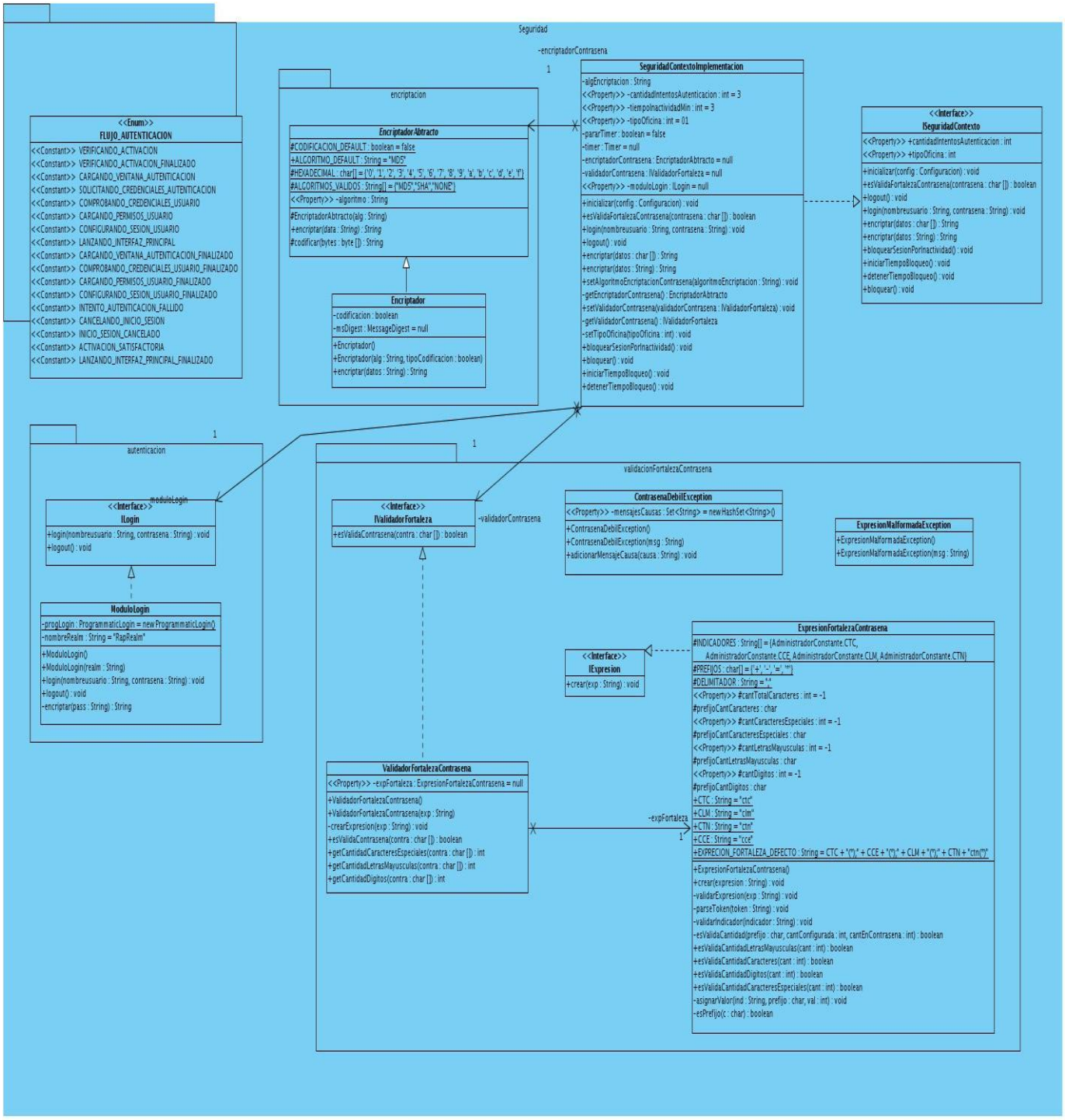


Diagrama de clases del paquete de componente de Seguridad

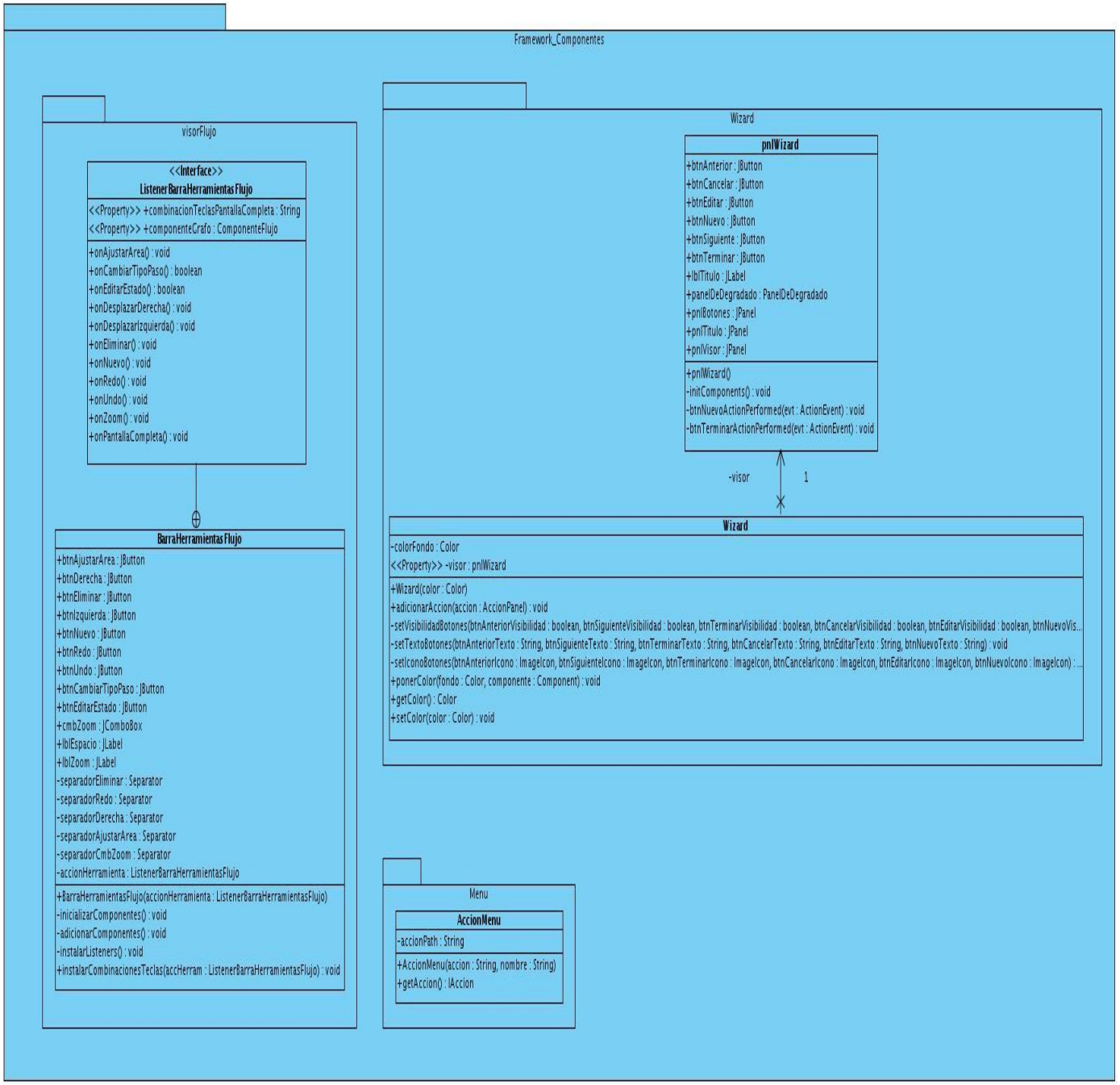


Diagrama de clases del paquete componente del framework

2.5.2 Diagramas de Clases del Framework para la parte del Servidor

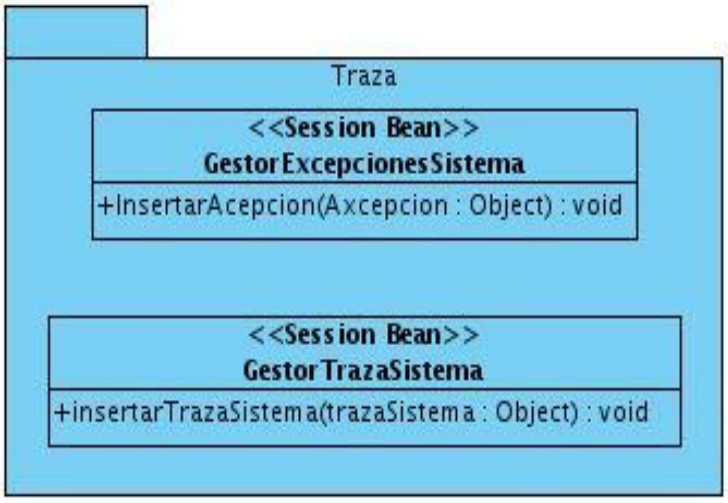


Diagrama de clases del componente Mensaje

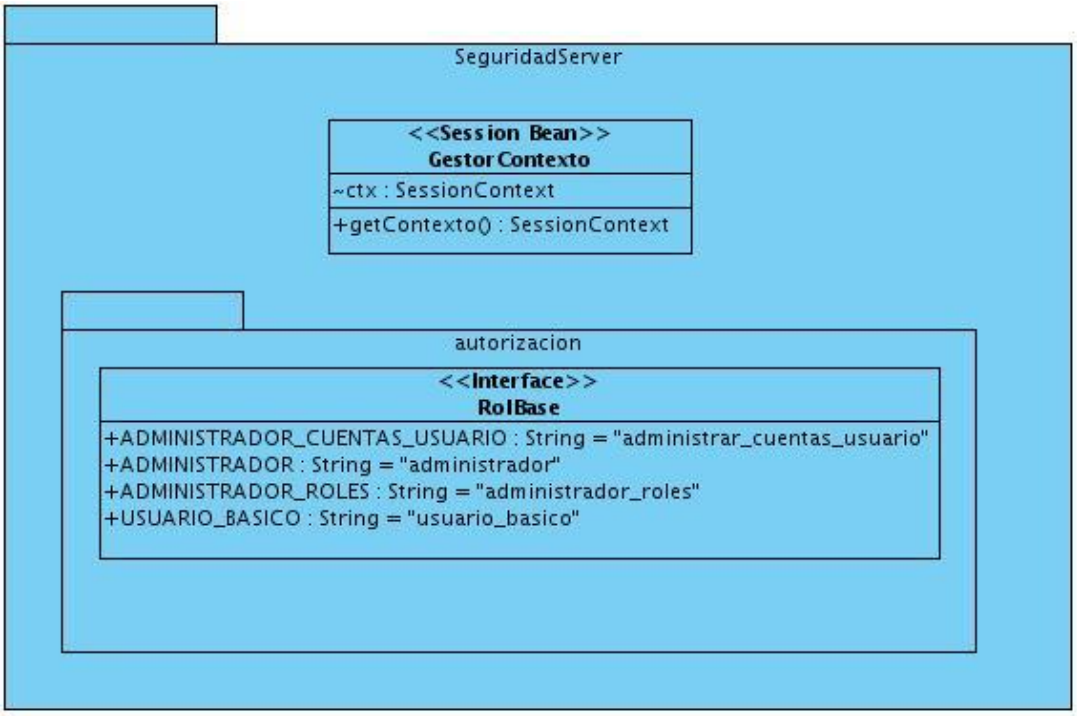


Diagrama de clases del componente de Seguridad.

2.5.3 Diagrama de Clases de los Casos de Uso

Gestionar Splash

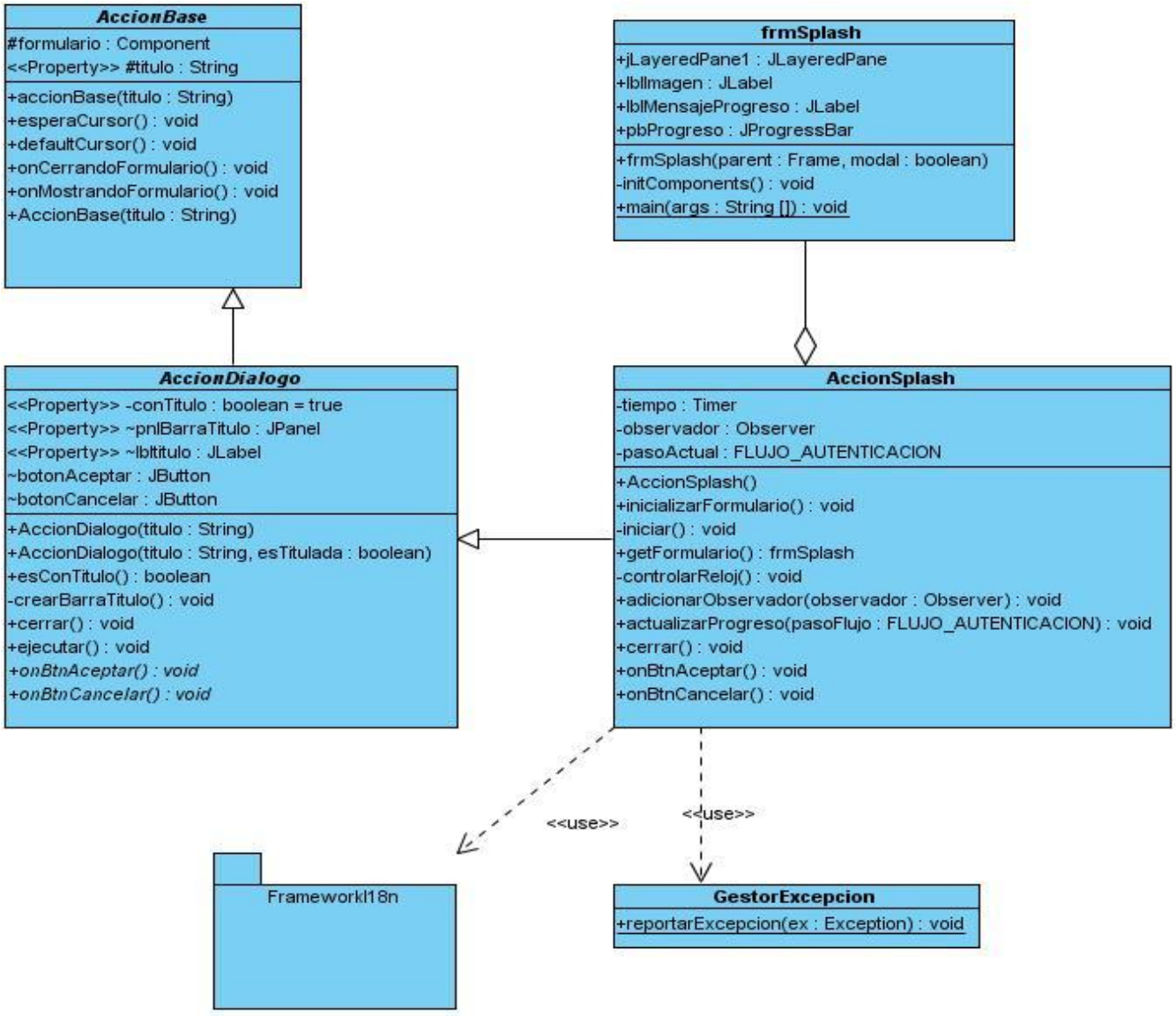


Diagrama del CU Gestionar Splash

Buscar Persona

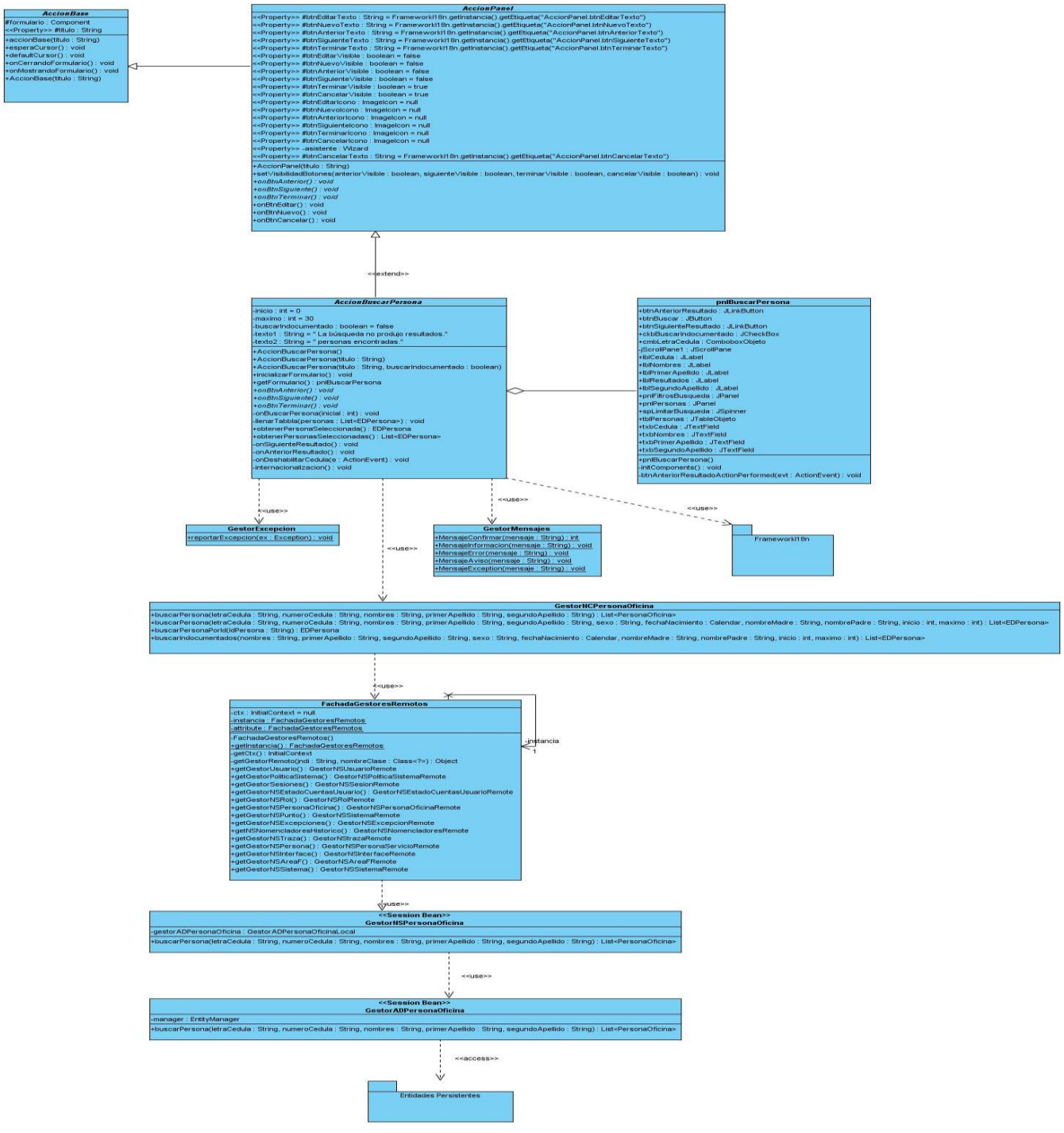


Diagrama del CU Buscar Persona

Gestionar Principal

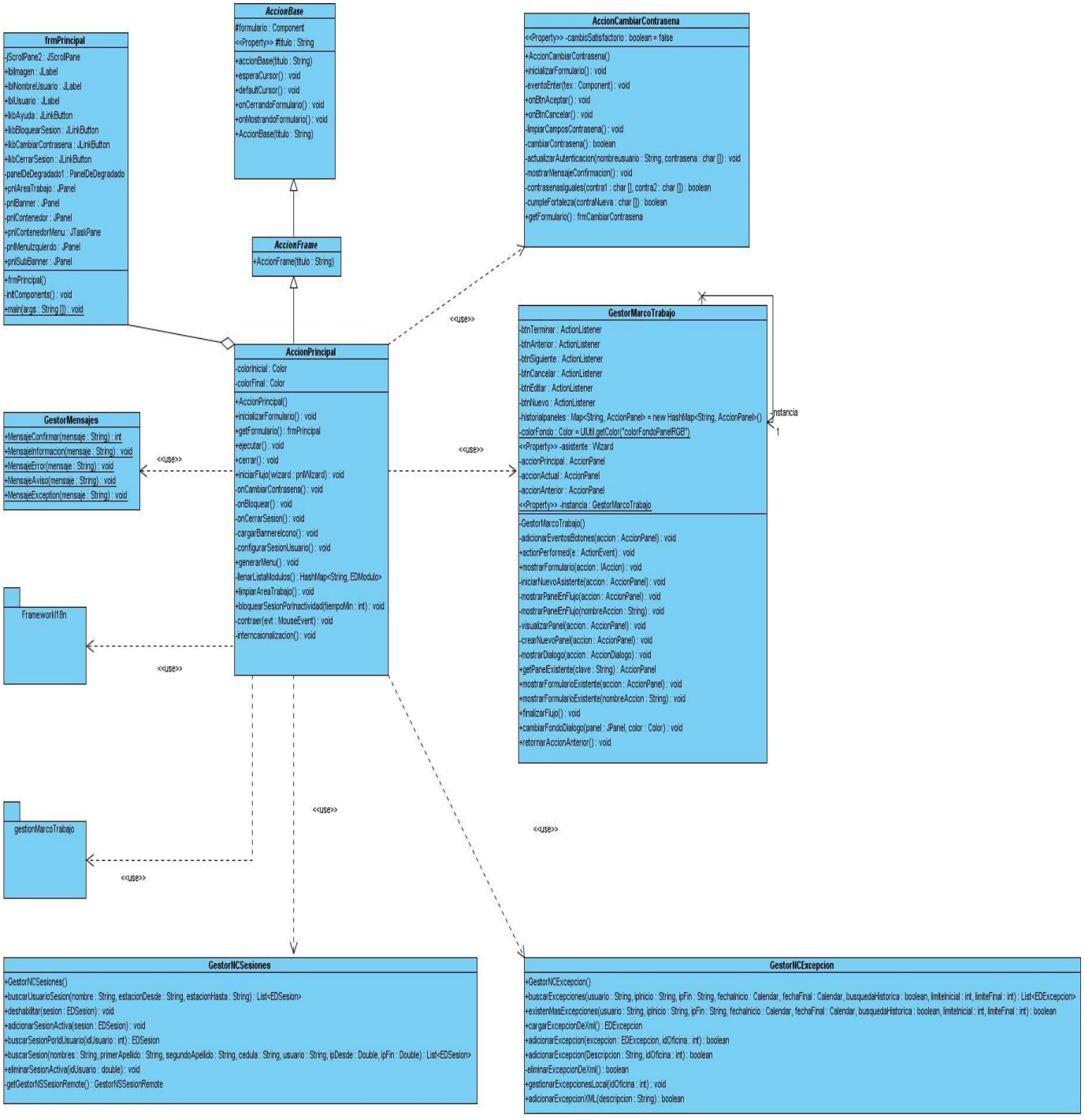


Diagrama del CU Gestionar Principal

Gestionar Mensajes Usuario

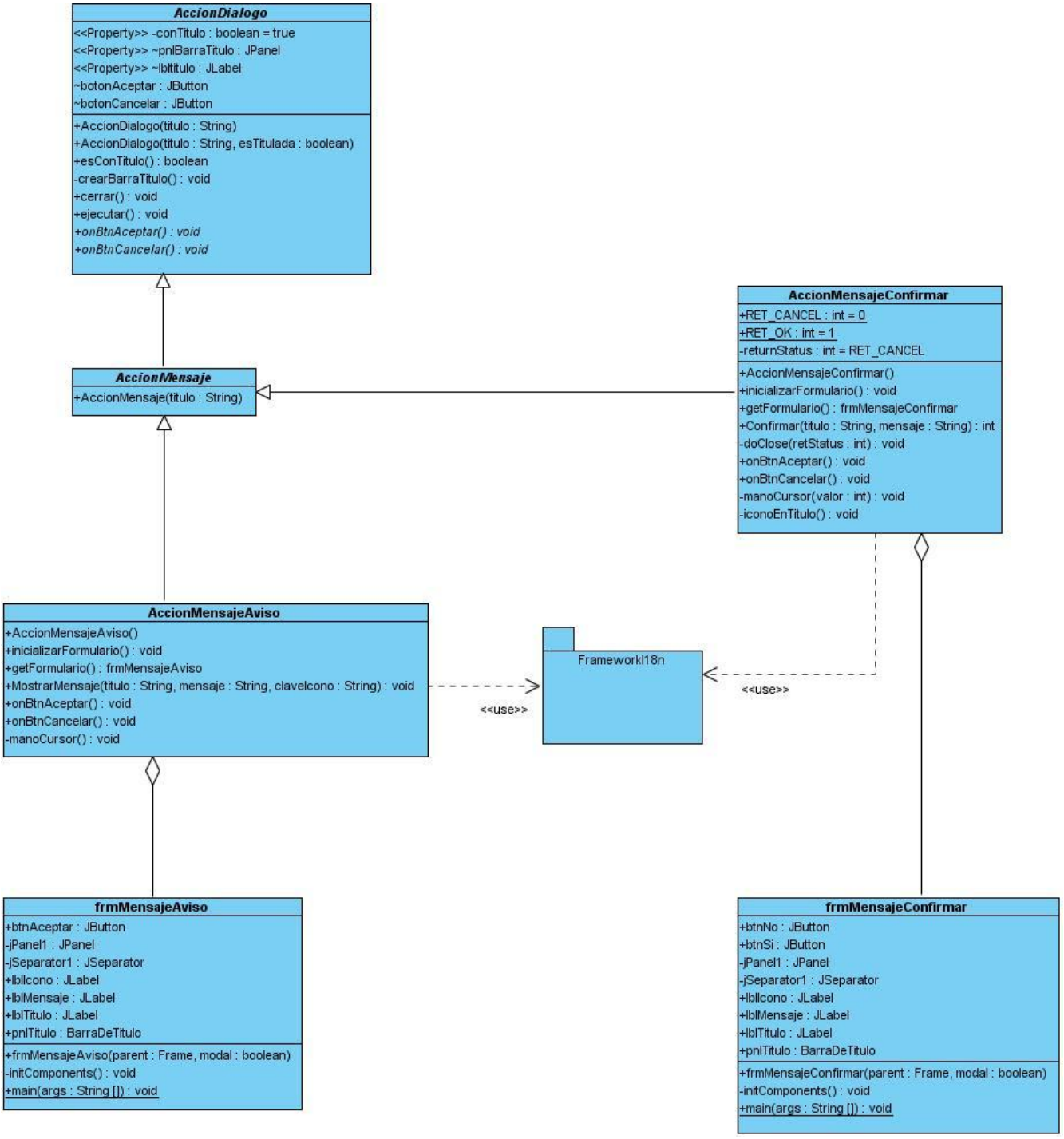


Diagrama del CU Gestionar Mensajes de Usuario

2.6 Diagrama de clases de las acciones del sistema.

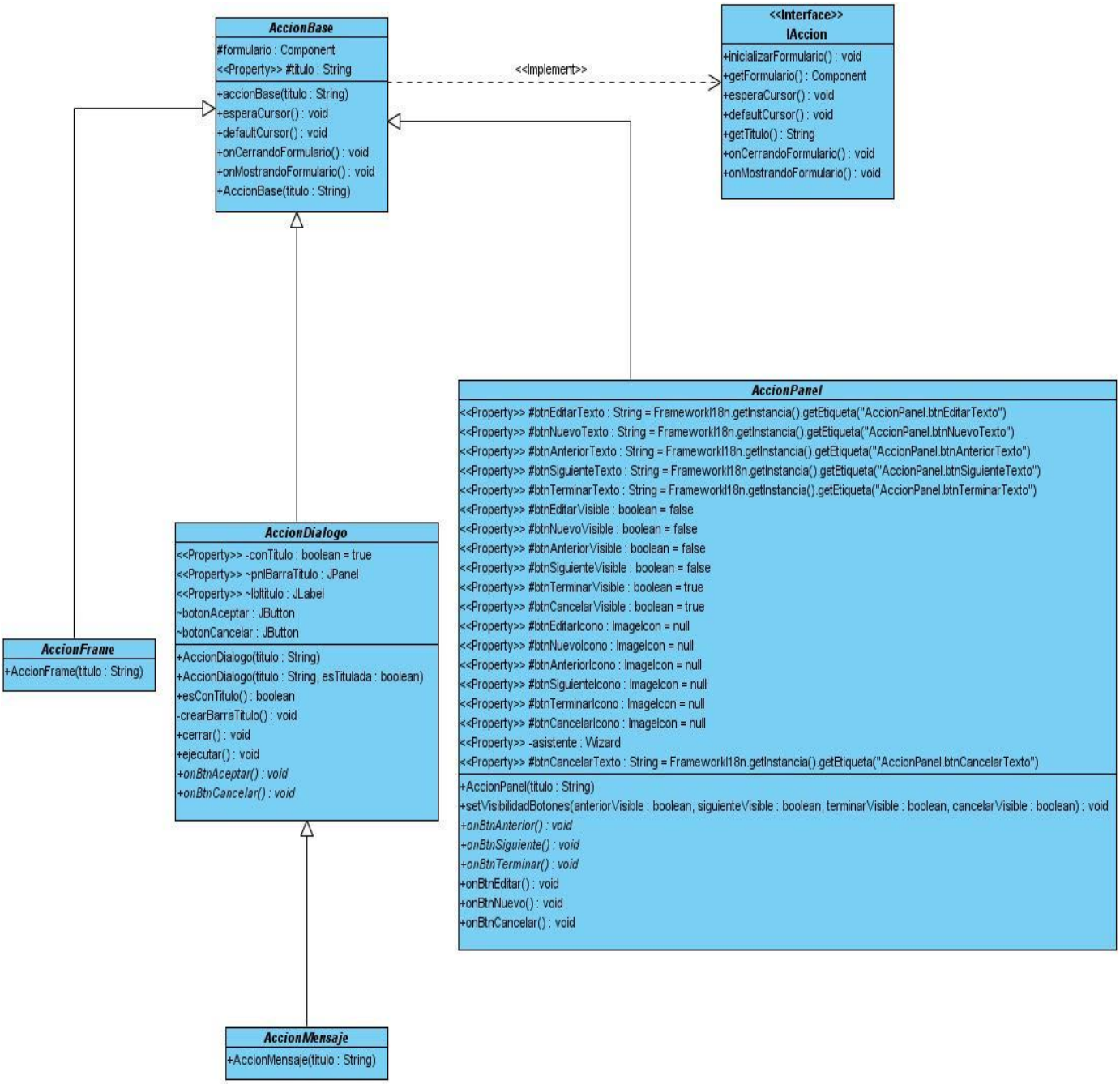


Diagrama de clases de las acciones del sistema.

2.7 Diagrama de los Paquetes del Sistema

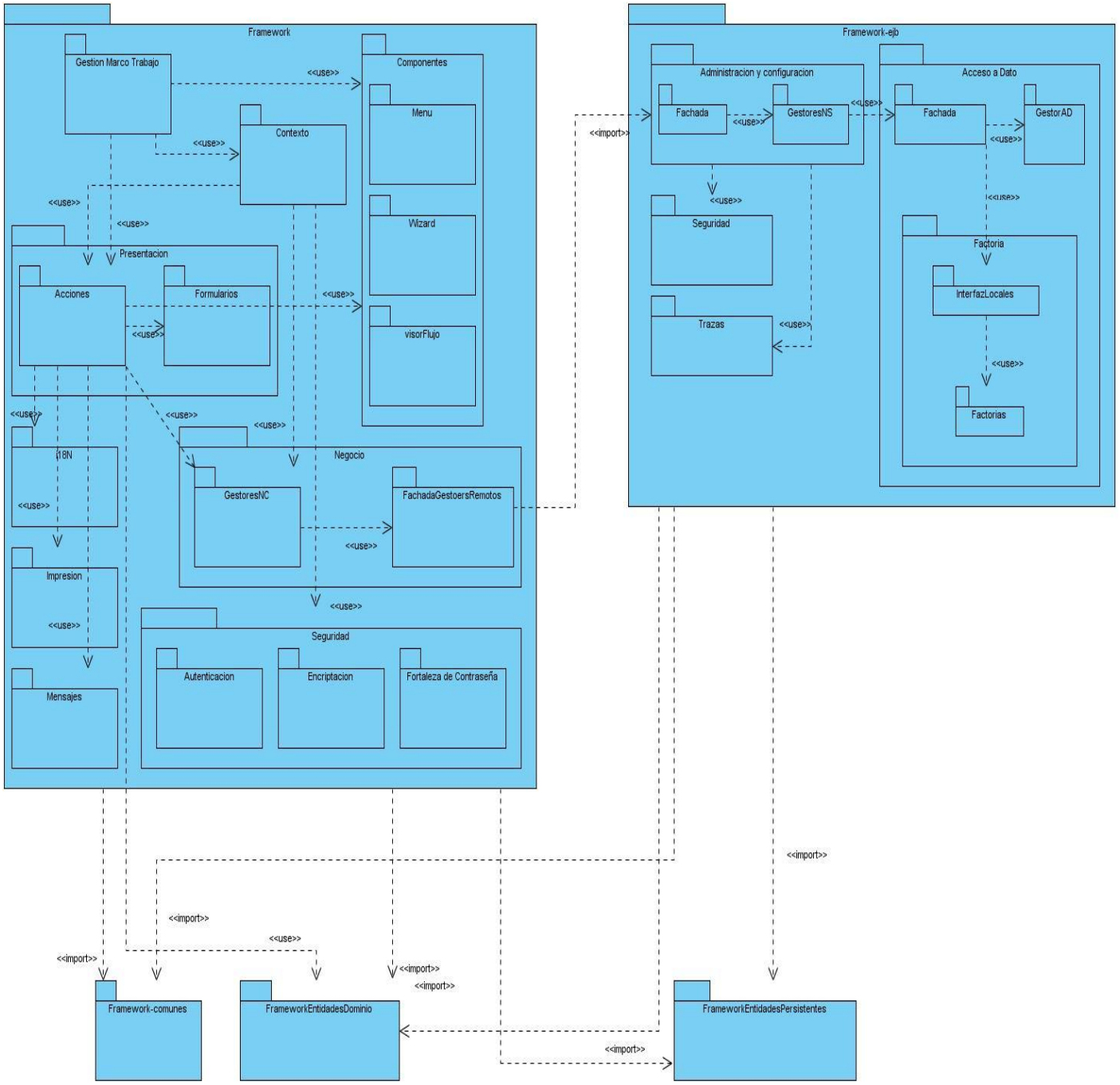


Diagrama de los Paquetes del Sistema

2.8 Descripción de Clases y Métodos del Paquete de Seguridad

Nombre	SeguridadContextoImplementacion
Tipo de Clase:	Controladora.
Descripción	
Se encarga de los métodos relacionados con la seguridad del sistema.	
Métodos	**Descripción**
esValidaFortalezaContrasena()	Valida la fortaleza de la contraseña.
logout()	Permite al usuario desconectarse.
login()	Permite al usuario conectarse.
encriptar()	Encripta la contraseña.
bloquearSesionPorInactividad()	Bloquea la sesión por inactividad.
getCantidadIntentosAutenticacion()	Obtiene la cantidad de intentos de autenticación.
iniciarTiempoBloqueo()	Comienza a contar el tiempo para el bloqueo por inactividad.
detenerTiempoBloqueo()	Detiene el tiempo para el bloqueo por inactividad.
bloquear()	Permite al usuario bloquear su sesión.

Nombre	ExpresionFortalezaContrasena
Tipo de Clase:	Controladora.
Descripción	
Se encarga de validar la fortaleza de la contraseña	
Métodos	**Descripción**
getCantCaracteresEspeciales()	Obtiene la cantidad de caracteres especiales que tiene la contraseña.

getCantDigitos()	Obtiene la cantidad de dígitos que tiene la contraseña.
getCantLetrasMayusculas()	Obtiene la cantidad de letras mayúsculas que tiene la contraseña.
getCantTotalCaracteres()	Obtiene la cantidad total de caracteres que tiene la contraseña.
esValidaCantidadLetrasMayusculas(int cant)	Valida que sea correcta la cantidad de letras mayúsculas que posee la contraseña.
esValidaCantidadCaracteres(int cant)	Valida que sea correcta la cantidad de caracteres que posee la contraseña.
esValidaCantidadDigitos(int cant)	Valida que sea correcta la cantidad de dígitos que posee la contraseña.
esValidaCantidadCaracteresEspeciales(int cant)	Valida que sea correcta la cantidad de caracteres especiales que posee la contraseña.

Nombre	ExpresionMalformadaException
Tipo de Clase:	Controladora.
Descripción	
Se encarga de enviar un mensaje si la contraseña está mal formada.	
Métodos	**Descripción**
ExpresionMalformadaException(String msg)	Se encarga de enviar un mensaje de error si la contraseña está mal formada.

Nombre	ValidadorFortalezaContrasena
Tipo de Clase:	Controladora.
Descripción	
Se encarga de validar la fortaleza de la contraseña	

Métodos	Descripción
esValidaContrasena(char[] contra)	Valida si es correcta o no la contraseña.
getCantidadDigitos(char[] contra)	Obtiene la cantidad de dígitos que tiene la contraseña.
getCantidadLetrasMayusculas(char[] contra)	Obtiene la cantidad de letras mayúsculas que tiene la contraseña.
getCantidadCaracteresEspeciales(char[] contra)	Obtiene la cantidad total de caracteres especiales que tiene la contraseña.
getExpFortaleza()	Obtiene la expresión de la contraseña.

Nombre	GestorContexto
Tipo de Clase:	Servidora.
Descripción	
Clase que facilita el contexto de sesión en el servidor Glassfish.	

Métodos	Descripción
getContexto()	Devuelve el contexto de una sesión.

Nombre	ILogin
Tipo de Clase:	Interface.
Descripción	
Esta clase realiza el proceso de autenticar un usuario.	

Métodos	Descripción
login()	Este método realiza el proceso de autenticar un usuario contra el mecanismo de autenticación del servidor de aplicaciones GlassFish.

logout()	Desconecta al usuario activo.

Nombre	EncriptadorAbtracto
Tipo de Clase:	Controladora.
Descripción	
Clase encargada de encriptar y codificar la contraseña.	
Métodos	**Descripción**
encriptar()	Método abstracto que encripta una cadena pasada.
codificar()	Método que codifica la cadena encriptada.
getAlgoritmo()	Retorna el tipo de algoritmo que se utilizará para encriptar.

Nombre	Encriptador
Tipo de Clase:	Controladora.
Descripción	
Clase encargada de encriptar la contraseña.	
Métodos	**Descripción**
encriptar()	Método que encripta la cadena pasada.

2.9 Descripción de las Clases y Métodos del CU Buscar Persona

Nombre	pnlBuscarPersona
Tipo de Clase:	Interfaz.

Descripción	
Clase que representa el formulario *BuscarPersona* en el que se introducen los datos necesarios para encontrar las Personas.	
Atributos	**Tipo**
btnAnteriorResultado	JLinkButton
btnBuscar	JButton
btnSiguienteResultado	JLinkButton
ckbBuscarIndocumentado	JCheckBox
cmbLetraCedula	ComboboxObjeto
jScrollPane1	JScrollPane
lblCedula	JLabel
lblNombres	JLabel
lblPrimerApellido	JLabel
lblSegundoApellido	JLabel
lblResultados	JLabel
pnlFiltrosBusqueda	JPanel
pnlPersonas	JPanel
spLimitarBusqueda	JSpinner
tblPersonas	JTableObjeto
txbCedula	JTextField

txbNombres	JTextField
txbPrimerApellido	JTextField
txbSegundoApellido	JTextField

Nombre	AccionBuscarPersona
Tipo de Clase:	Controladora.
Descripción	
Clase Genérica que controla las funcionalidades del formulario BuscarPersona.	
Métodos	**Descripción**
onBtnAnterior()	Método abstracto que se implementa en las distintas clases que heredan de BuscarPersona.
onBtnSiguiente()	Método abstracto que se implementa en las distintas clases que heredan de BuscarPersona.
onBtnTerminar()	Método abstracto que se implementa en las distintas clases que heredan de BuscarPersona.
onBuscarPersona()	A partir de los parámetros entrados, puede ser uno o varios el método busca las personas asociadas.
llenarTabbla()	A partir de una lista de personas llena la tabla.
obtenerPersonaSeleccionada()	Devuelve los datos de la persona seleccionada.
onSiguienteResultado()	Muestra la opción de escoger la cantidad de resultados a mostrar en la tabla siguiente.
onAnteriorResultado()	Muestra la opción de escoger la cantidad de resultados a mostrar en la tabla anterior.
onDeshabilitarCedula()	Deselecciona el campo cédula en la tabla.

internacionalizacion()	Crea un atributo del tipo frmPrincipal y añade las etiquetas del paquete i18N.

Nombre	GestorMensajes
Tipo de Clase:	Controladora.
Descripción	
Clase que agrupa todos los mensajes que pueda lanzar el sistema.	
Métodos	**Descripción**
MensajeConfirmar()	Método que lanza un mensaje de confirmación dado un mensaje pasado por parámetro.
MensajeInformacion()	Método que lanza un mensaje de información dado un mensaje pasado por parámetro.
MensajeError()	Método que lanza un mensaje de error dado un mensaje pasado por parámetro.
MensajeAviso()	Método que lanza un mensaje de aviso dado un mensaje pasado por parámetro.
MensajeException()	Método que lanza un mensaje de excepción dado un mensaje pasado por parámetro.

Nombre	GestorExcepcion
Tipo de Clase:	Controladora.
Descripción	
Clase que agrupa todas las excepciones que pueda lanzar el sistema.	
Métodos	**Descripción**
reportarExcepcion()	Agrupa las excepciones lanzadas por el sistema.

run()	Crea un nuevo grupo de excepción.

Nombre	GestorNCPersonaOficina
Tipo de Clase:	Controladora.
Descripción	
Clase que gestiona la Lógica de Negocio del Cliente para la entidad PersonaOficina que hereda de EDPersona. Realiza las llamadas correspondientes a la Lógica de Negocio del Servidor para completar las funcionalidades.	
Métodos	**Descripción**
buscarPersonaPorId()	Retorna la persona que coincide con el id de persona pasado como parámetro.
buscarIndocumentados()	Retorna una lista de personas encontradas bajo los parámetros pasados.
buscarPersona()	Método que hereda de la clase AccionBuscarPersona y retorna todos los resultados encontrados de persona.

Nombre	GestorNSPersonaOficina
Tipo de Clase:	Controladora.
Descripción	
Clase que se utiliza para gestionar la lógica de negocio del servidor para la entidad PersonaOficina que hereda de EDPersona.	
Métodos	**Descripción**
buscarPersona()	Busca y Retorna una lista de Personas que concuerdan con los atributos pasados como parámetro.

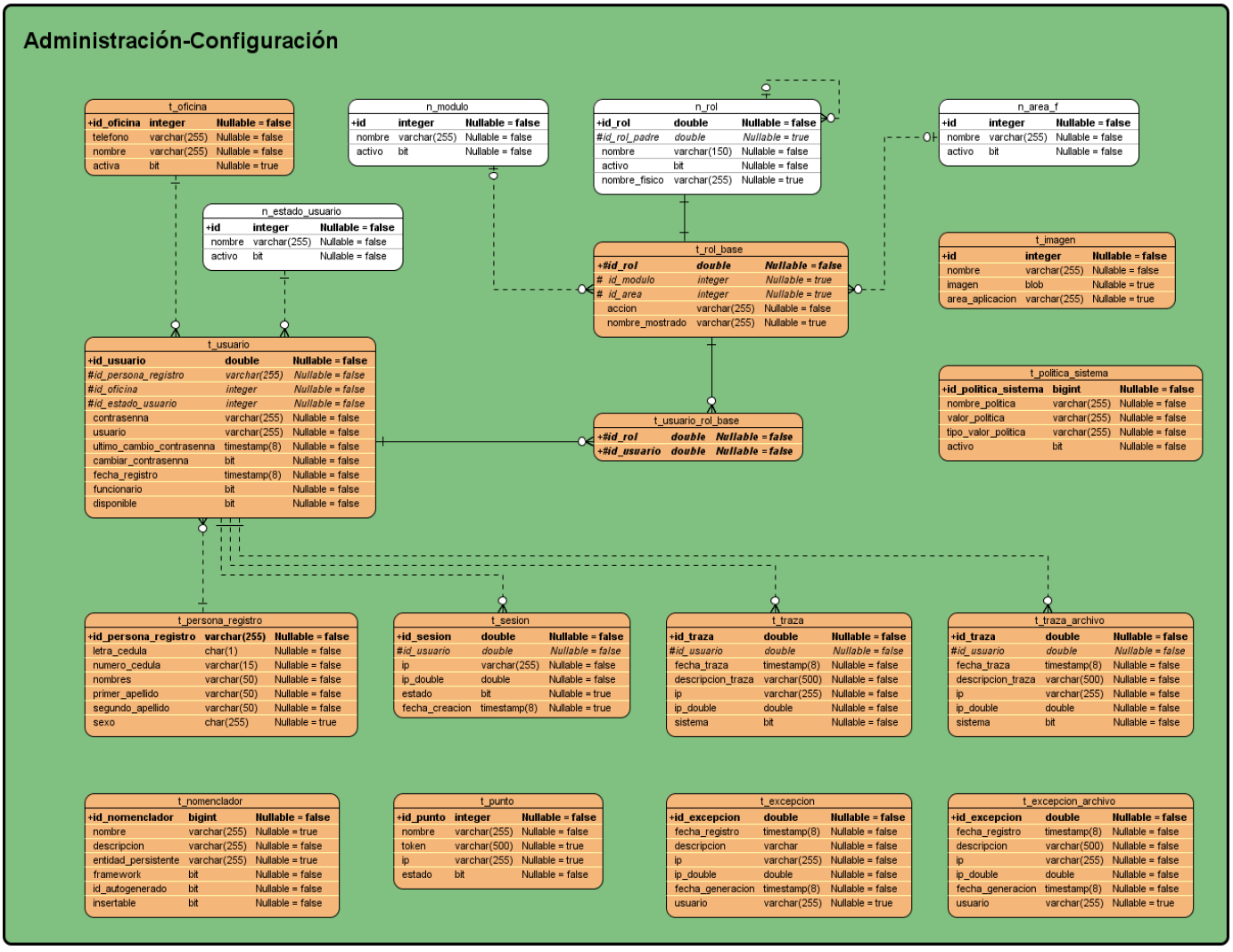
Nombre	GestorADPersonaOficina
Tipo de Clase:	Controladora.
Descripción	
Clase que se utiliza para gestionar la lógica de acceso a datos para la entidad Sesión.	
Métodos	**Descripción**
buscarPersona ()	Devuelve una lista de personas.

Las restantes descripciones de las clases se encuentran en el anexo1 del documento.

2.10 Modelo de Datos

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, un modelo de datos permite describir las estructuras de la base de datos, las restricciones de integridad y las operaciones de manipulación. En un enfoque más amplio, un modelo de datos permite describir los elementos que intervienen en una realidad o en un problema dado y la forma en que se relacionan dichos elementos entre sí.

Modelo de Datos del Marco de Trabajo



Resumen

Nombre	Descripción
t_oficina	Entidad que almacena los datos relacionados con cada oficina del local de la división.
n_modulo	Nomenclador que representa todos los módulos presentes en el sistema.

n_rol	Representa todos los roles que existentes en el sistema.
n_area_f	Nomenclador que almacena los datos relacionados a las áreas existentes.
n_estado_usuario	Nomenclador que almacena todos los estados de los usuarios.
t_imagen	Nomenclador que almacena los datos relacionados a las imágenes existentes.
t_rol_base	Almacena los datos generales que representan los roles desempeñados por los usuarios del sistema.
t_usuario	Almacena los datos generales de los usuarios del sistema.
t_politica_sistema	Tabla que almacena los datos de configuración de las políticas del sistema.
t_usuario_rol_base	Representa la relación entre la tabla usuario y rol_base, especificando que los usuarios pueden tener muchos roles base y los roles base, pueden ser desempeñados por varios usuarios.
t_persona_registro	Representa los datos de todas las personas que se encuentran registradas en el sistema que pueden ser usuarios del sistema.
t_sesion	Almacena las sesiones presentes en el sistema.
t_traza	Almacena las trazas de forma general que surgen en el sistema.
t_traza_archivo	Almacena todas las trazas existentes en el sistema.
t_nomenclador	Entidad que almacena de forma general los nomencladores del sistema.
t_punto	Almacena todos los puntos de los puestos de trabajo en el sistema.
t_excepcion	Almacena de forma general las excepciones o errores que puedan ocurrir o dar el sistema.
t_excepcion_archivo	Almacena de forma general las excepciones o errores que puedan ocurrir o dar el sistema.

2.11 Conclusiones

Durante el transcurso de este capítulo se logró una visión general de las principales características del Marco de Trabajo, partiendo de los requisitos funcionales. Además, se obtuvieron varios artefactos importantes como son el diagrama de clases de cada uno de los componentes, el diagrama de paquetes, y el modelo de datos.

Capitulo 3: Implementación

La implementación comienza con el resultado del diseño, se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables, entre otros. La implementación está fuertemente determinada por el lenguaje de programación.
Este capítulo describe el modelo de implementación y aborda temas importantes como son el tratamiento de excepciones, la seguridad del sistema, y los estándares de codificación.

3.1 Modelo de Implementación

Está conformado por los Diagramas de Componentes y de Despliegue, describiendo cómo los elementos del Modelo de Diseño se implementan en términos de componentes, ficheros de código fuente y ejecutables. El Modelo de Implementación describe además cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización, disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y cómo dependen los componentes unos de otros.

3.1.1 Diagrama de Componentes

Es una representación gráfica que muestra un conjunto de elementos del modelo tales como componentes, subsistemas o paquetes de implementación y sus relaciones. Permite modelar la vista estática del sistema mostrando las dependencias lógicas entre un conjunto de componentes de software. Este diagrama se estructura en paquetes, que son divisiones físicas del sistema. Los paquetes están organizados en una jerarquía de capas donde cada capa tiene una interfaz bien definida. Los componentes pueden ser de código fuente, librerías, binarios o ejecutables y tienen relaciones de traza con los elementos del modelo que implementan. A continuación se muestra el diagrama de componentes dividido por capas.

Capa de Presentación

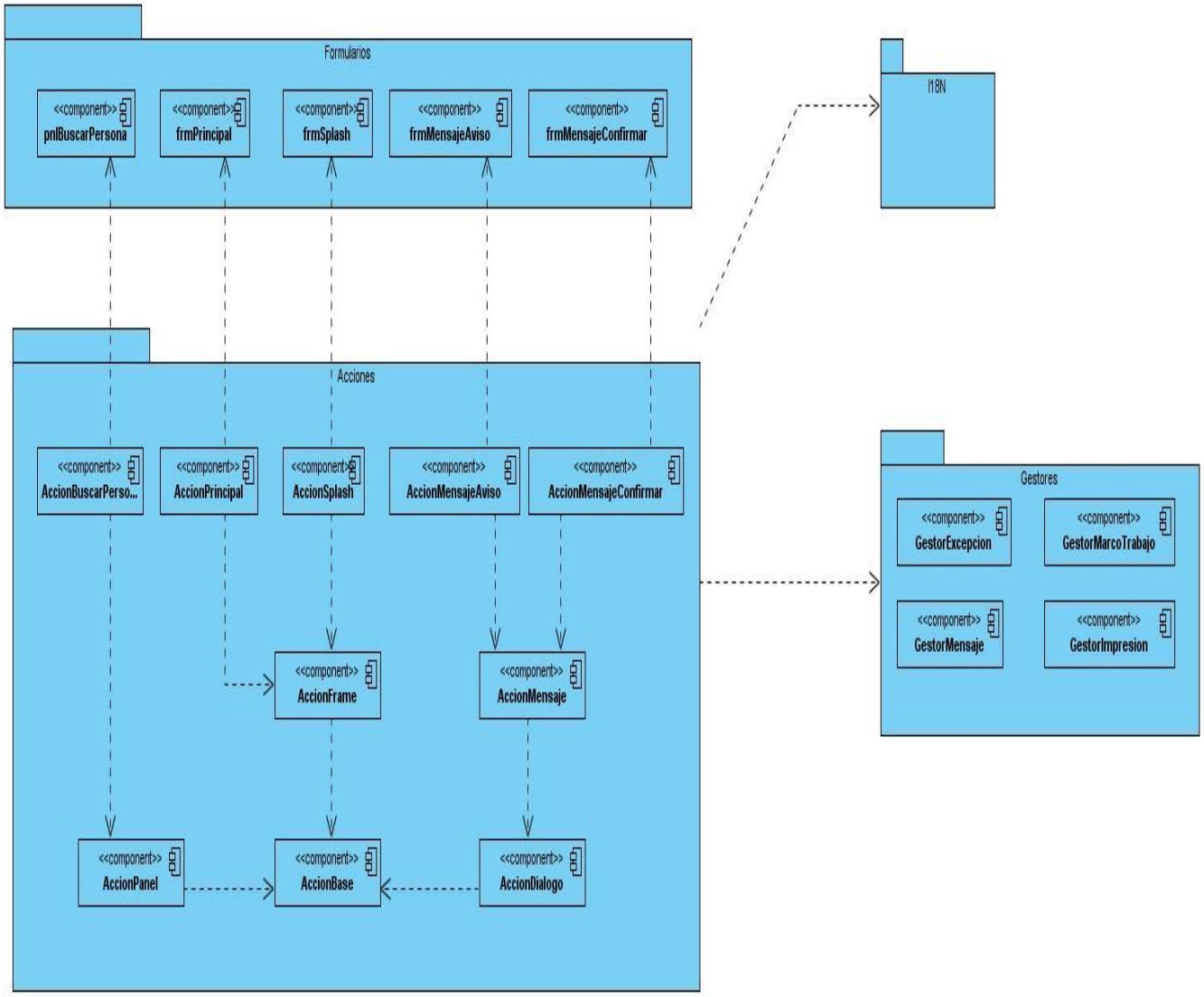


Diagrama de componentes de la capa de presentación.

Capa de Lógica de Negocio

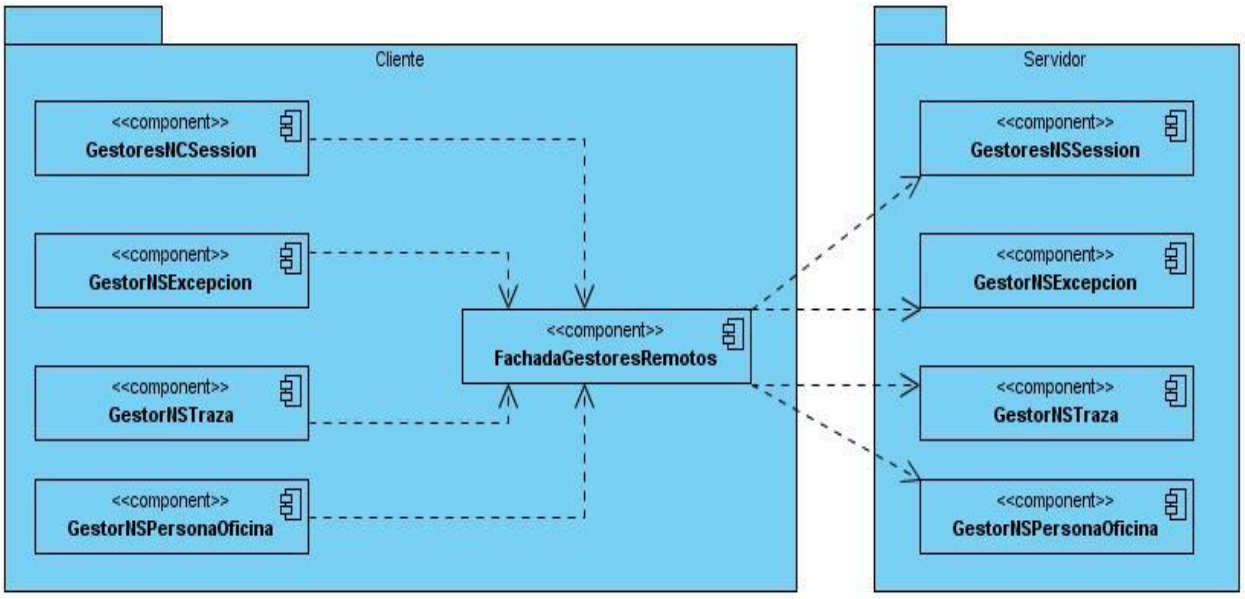


Diagrama de componentes de la capa de lógica de negocio

Capa de Acceso a Datos

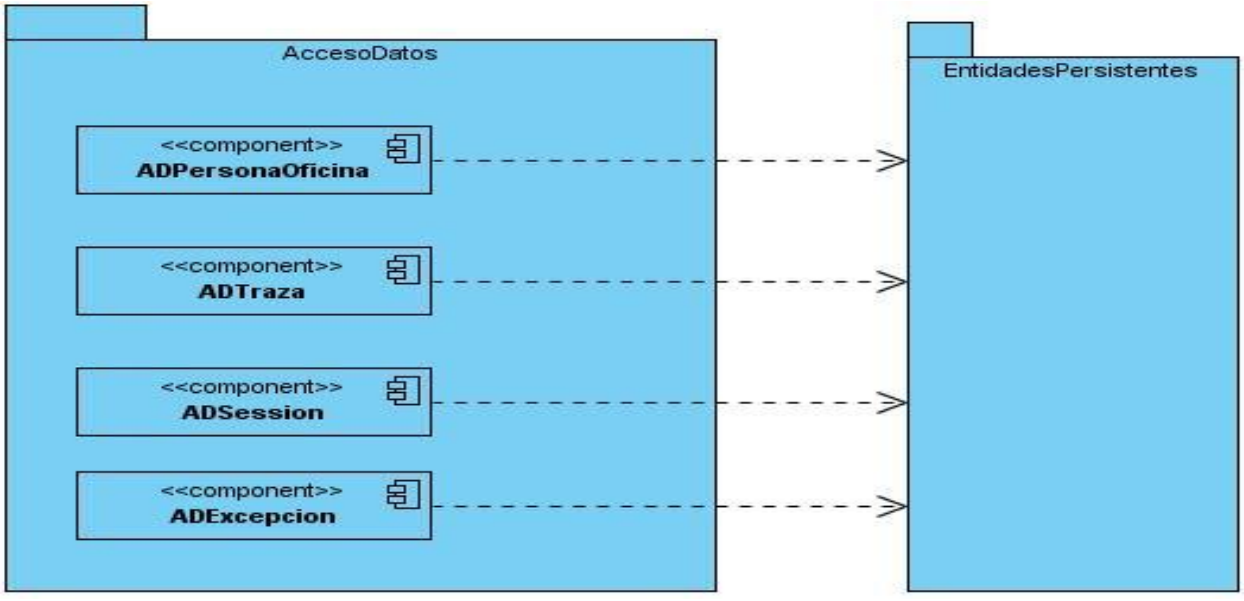


Diagrama de componentes de la capa de acceso a datos.

3.1.2 Diagrama de Integración del Marco de Trabajo con los Subsistemas

A continuación se muestra un diagrama de integración de forma general donde se puede apreciar la interacción del Marco de Trabajo con los subsistemas Centro de Digitalización y Solución de Gestión.

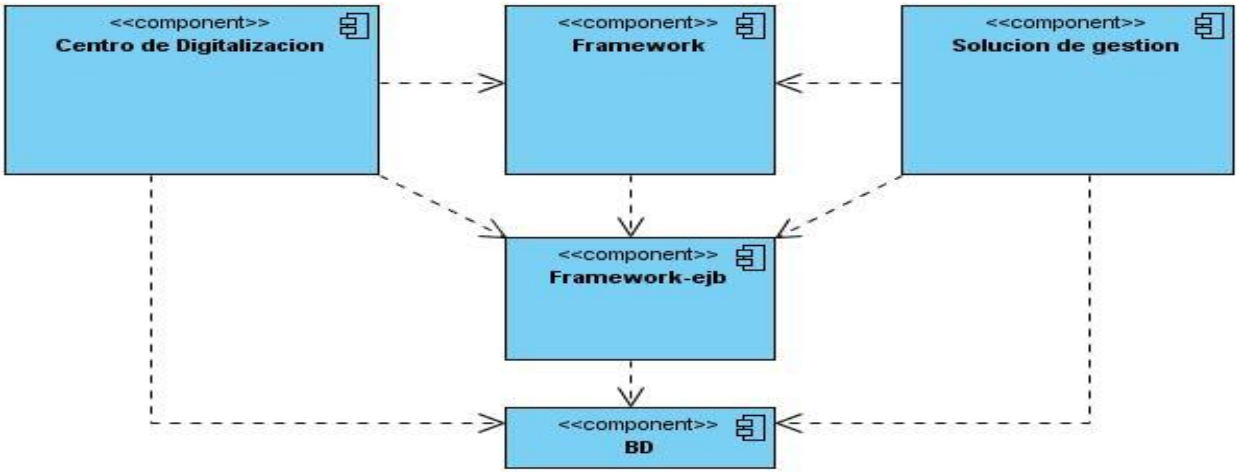


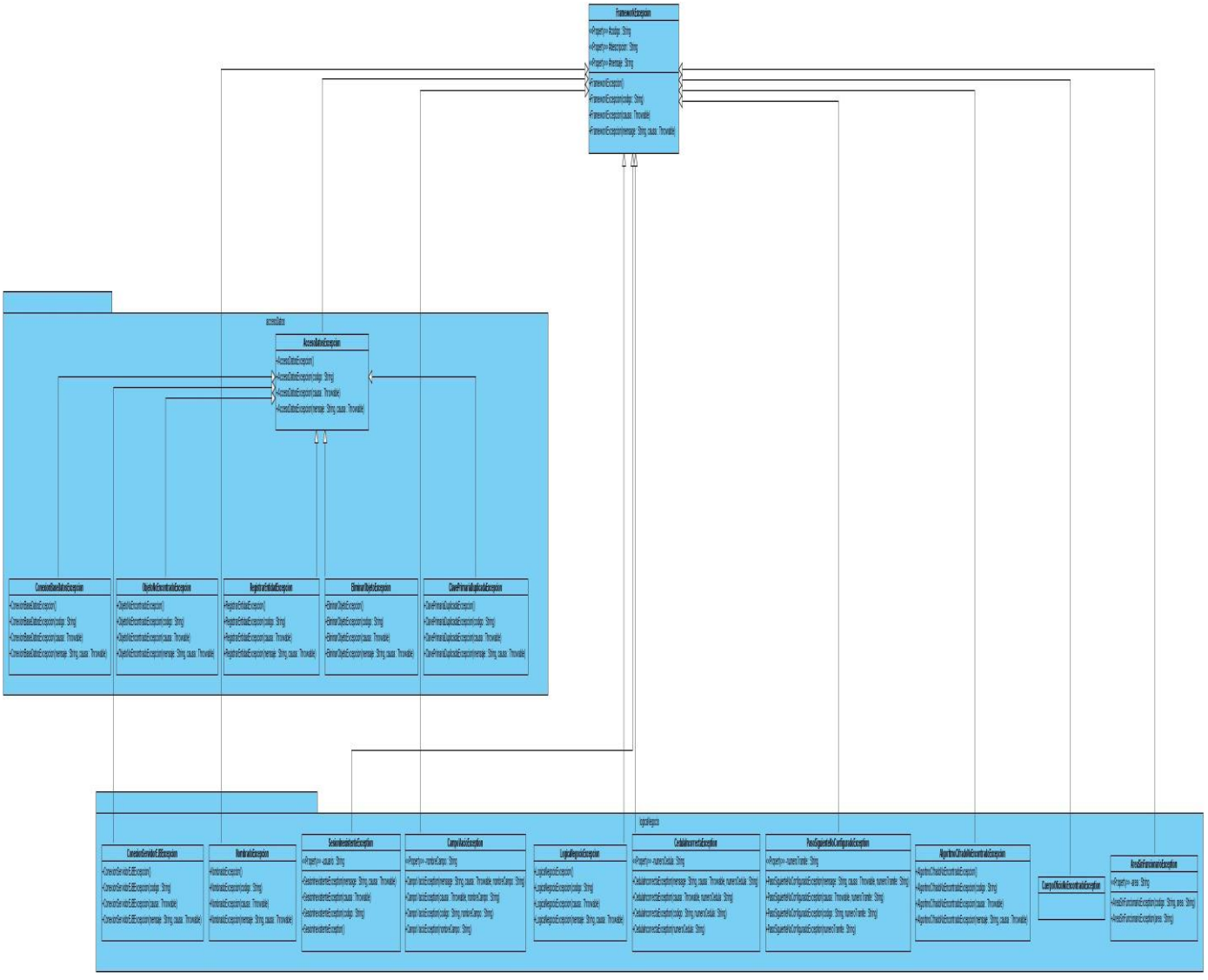
Diagrama de integración del Marco de Trabajo con los subsistemas

3.2 Tratamiento de excepciones

Una excepción es un evento que ocurre durante la ejecución de un programa interrumpiendo el flujo normal de las sentencias. Dicho evento puede ser desde serios problemas de hardware, como la avería de un disco duro, hasta los simples errores de programación y pueden ser tratados mediante una estructura de control que poseen los lenguajes de programación de alto nivel, diseñada para manejar condiciones anormales que pueden ser tratadas por el mismo programa que se desarrolla. A esta estructura de control se le conoce como tratamiento de excepciones. Un buen tratamiento de excepciones permite el aumento de la calidad del software, además de obtener un sistema más robusto y fiable.

En el sistema a desarrollar el control de las excepciones se llevará a cabo en todo el código, donde pueda surgir alguna situación inesperada, especialmente donde se ejecutan sentencias que manipulan los datos que viajan desde y hacia la base de datos. También se controlan los errores que pueden surgir en validación de datos provenientes de la interfaz de usuario. En las clases controladoras de procesos, se utilizará el bloque try para detectar cuando ocurra algún fallo y mediante el catch se manejarán dichas

excepciones, mediante mensajes que se muestran en la interfaz de usuario. Existe además un archivo XML, que engloba la configuración de todos los mensajes que se deben mostrar por cada tipo de excepción, así como la página a la que el sistema redirecciona en caso de la aparición de un error sorpresivo. También se personalizaron algunas Excepciones para un mejor entendimiento del grupo de desarrollo y una mayor comodidad.



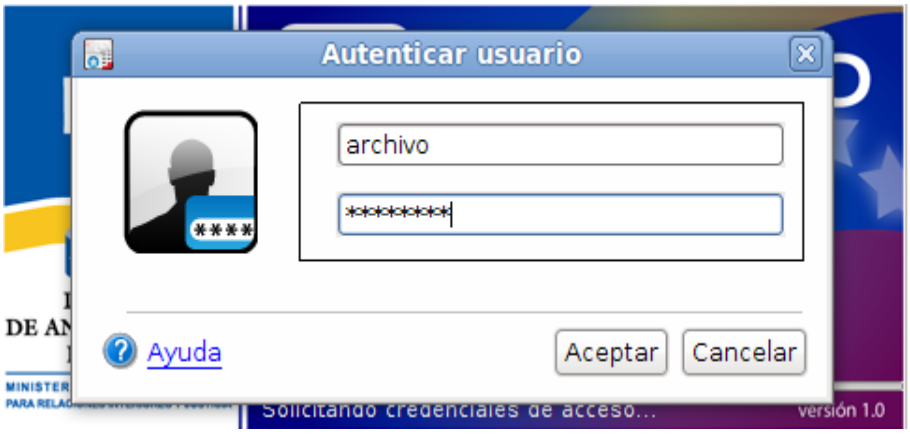
Estructura jerárquica de las excepciones.

3.3 Seguridad

La seguridad es un aspecto de gran importancia y se garantiza cuando se cumplen los aspectos de integridad, confidencialidad y disponibilidad. Para lograr esto fue necesario trabajar en dos aspectos fundamentales, la autenticación, y la autorización. La primera fue implementada en la parte del cliente, que es la encargada de la interacción con el usuario, mientras que la segunda fue implementada en la parte del servidor. La unión de estas características trae consigo que la información pueda ser vista y modificada solo por el personal autorizado; de forma controlable y que esté disponible cuando se necesite. A continuación se describen detalladamente las funcionalidades que ofrece el Marco de Trabajo, para garantizar la seguridad del sistema.

Autenticación

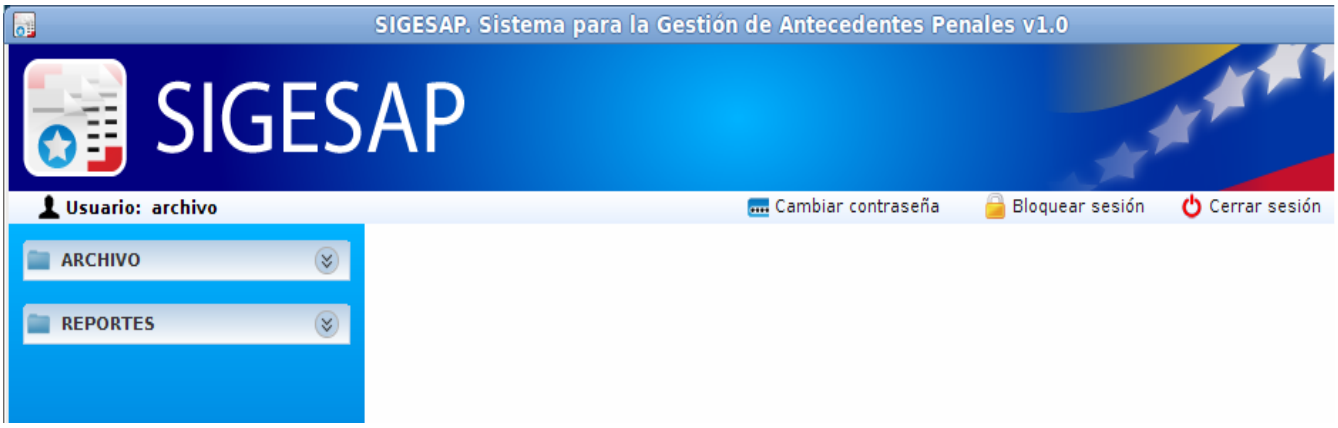
La autenticación es la parte encargada del acceso al sistema y para ello se tomaron en cuenta varios aspectos importantes como son la encriptación, la fortaleza de la contraseña, y la cantidad de intentos. Para la encriptación se utilizo el algoritmo MD5 el cual es un algoritmo de reducción criptográfico de 128 bits ampliamente usado, y que es representado típicamente como un número de 32 dígitos hexadecimal. Para la fortaleza de la contraseña se tuvo en cuenta el número total de caracteres, la cantidad de números, la cantidad de caracteres especiales y la cantidad de letras mayúsculas que la misma debiese poseer. Otro aspecto importante fue la cantidad de intentos para la autenticación, en aras de evitar posibles ataques de fuerza bruta.



Autenticación del sistema

Autorización

La autorización es la parte encargada del acceso a las funcionalidades del sistema, en dependencia del rol del usuario autenticado, garantizando que solo pueda acceder a la información el personal autorizado. Para esto se hizo uso del glassfish como servidor de aplicaciones, el cual posee mecanismos para controlar los permisos de los usuarios mediante anotaciones.



Usuario archivo autenticado con el nivel de acceso correspondiente

Otra de las opciones que ofrece el sistema y que forman parte de la seguridad del mismo son:

- Cerrar sesión y salir del sistema.
- Bloquear la sesión de trabajo.
- Bloqueo de sesión por inactividad.
- Cambiar contraseñas.
- Almacenar trazas en la Base de Datos de las acciones que realicen los usuarios.

Opciones de seguridad del sistema

3.4 Estándares de Codificación

Los estándares de codificación definen un conjunto de reglas para escribir el código, los cuales permiten obtener un código de alta calidad y cumplir con las buenas prácticas establecidas en la Ingeniería de Software. Además posibilita que el software que se obtiene sea fácil de comprender. A continuación se describen los estándares de codificación que se utilizarán para el desarrollo del sistema.

3.4.1 Nomenclatura para los paquetes, métodos y clases

- Los paquetes (o package) deberán comenzar con el nombre del sistema (SIGESAP) seguido por el subsistema, luego el módulo y por último la capa lógica a la que pertenece.
- Se utilizará la notación Camello en los nombres de las Entidades persistentes, Entidades de Dominio, Propiedades y Funcionalidades del sistema.
- Las clases persistentes comenzarán con las siglas definidas en la BD para cada tabla.
- Los formularios de tipo popup comienzan con las siglas Frm, y los de tipo panel con las siglas Pnl.
- Los gestores de negocio del cliente comienzan con las siglas GestorNC y los del servidor con las siglas GestorNS.
- Las clases de la lógica de negocio de acceso a datos comienzan con las siglas GestorAD.
- Las interfaces publicadas en el servidor terminan con el prefijo Remoto.
- Las clases interceptoras de validación terminan con el prefijo Interceptor.
- Las excepciones terminan con el sufijo Exception.

3.4.2 Nomenclatura para los componentes de formularios

- Los botones (JButton) comienzan con las siglas btn.
- Los campos de texto (JTextField) comienzan con las siglas tfd.

- Los campos fecha (Date) comienzan con las siglas fch.
- Para el componente JDateChooser con las siglas dtc.
- Para el componente JDayChooser con las siglas dyc.
- Los componentes de tipo Jscrool usarán las siglas sc.
- Las cajas de texto (JComboBox) comienzan con las siglas cmb.
- Las áreas de texto (JTextArea) comienzan con las siglas txt.
- Los labels (JLabel) comienzan con las siglas lbl.
- Los cuadros de chequeo (JCheckBox) comienzan con las siglas chb.
- Los radiobutton (JRadioButton) comienzan con las siglas rbt.
- Las tablas (JTable) comienzan con las siglas tbl.

3.5 Conclusiones

El resultado fundamental de este capítulo fue la construcción del sistema, para el cual se utilizaron diferentes artefactos de implementación como el diagrama de componentes y el diagrama de integración. Además, se tuvo en cuenta aspectos importantes como son el tratamiento de excepciones, la seguridad, y los estándares de codificación establecidos para el desarrollo del Marco de Trabajo.

Capitulo 4: Validación de la solución

En la implementación de diferentes sistemas informáticos juega un papel esencial el uso de las técnicas de evaluación dinámica o pruebas. Para lograr esto se deben llevar a cabo estrategias a la hora de evaluar dinámicamente el software, pues para la evaluación de este, se debe comenzar desde los componentes más simples y pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, debe efectuarse en cada una de las etapas de desarrollo.

En este capítulo se muestran los casos de prueba y las métricas que se utilizaron para validar la solución y asegurar la calidad del producto.

4.1 Métricas de Software

Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento desde diferentes puntos de vista. (Briand et al., 1996)

En general, la medición persigue tres objetivos fundamentales (Fenton y Pfleeger, 1997):

- Entender qué ocurre durante el desarrollo y el mantenimiento
- Controlar qué es lo que ocurre en nuestros proyectos
- Mejorar los procesos y productos

Principales atributos para medir la calidad de un software

1. Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

2. Complejidad del diseño: Consiste en la complejidad que posee una estructura de diseño de clases.

3. Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

4. Reutilización: Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.

5. Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización. Complejidad del mantenimiento. Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

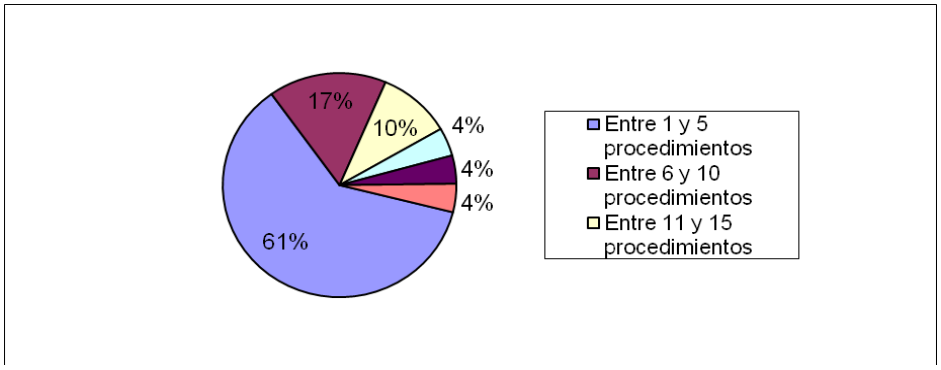
6. Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, modulo, clase, conjunto de clases, entre otros.) diseñado.

4.1.1 Tamaño operacional de clase (TOC)

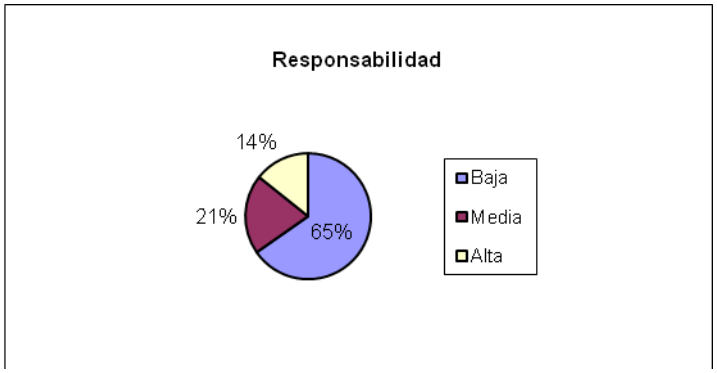
Esta métrica está relacionada con el número de métodos asignados a una clase. Teniendo en cuenta los siguientes atributos:

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la Responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

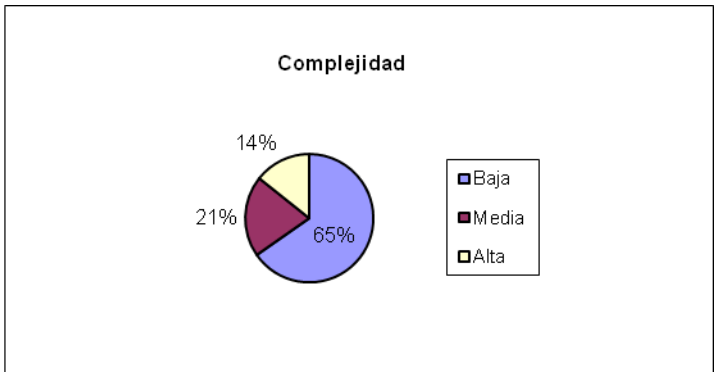
Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:



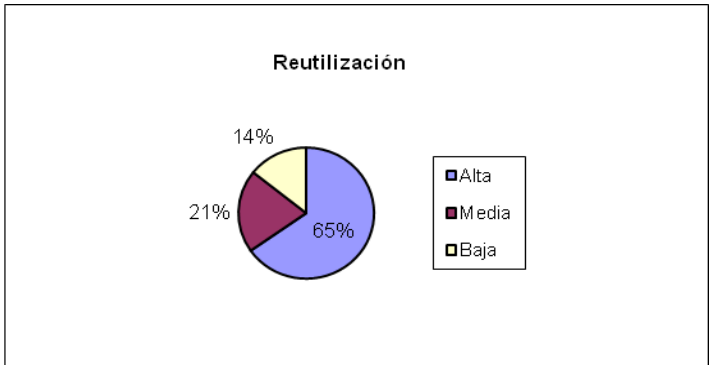
Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.



Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.



Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.



Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del Marco de Trabajo tienen una buena calidad pudiéndose observar que el 61% de las clases posee menos cantidad de operaciones que la media registrada en las mediciones. Además el 65% de las clases posee evaluaciones positivas en los atributos (Responsabilidad, Complejidad de Implementación y Reutilización).

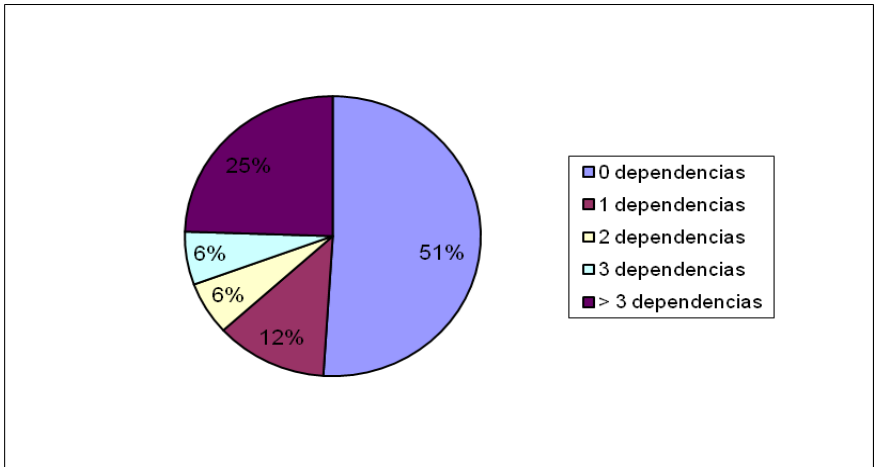
4.1.2 Relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otras.

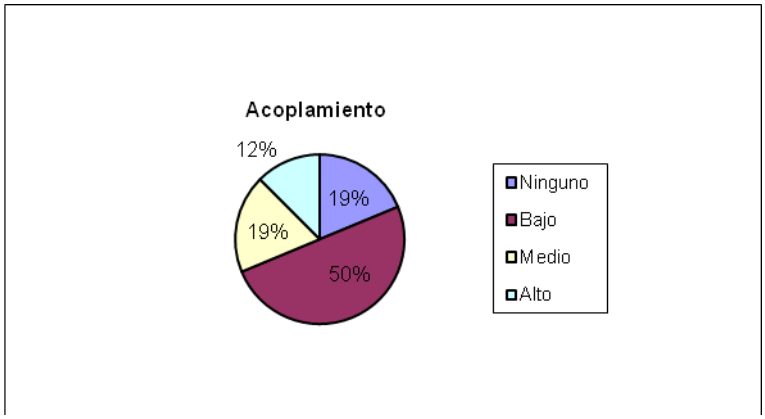
Atributo que afecta	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

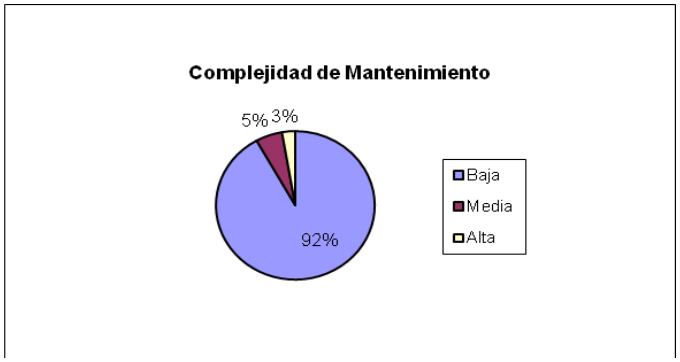
Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:



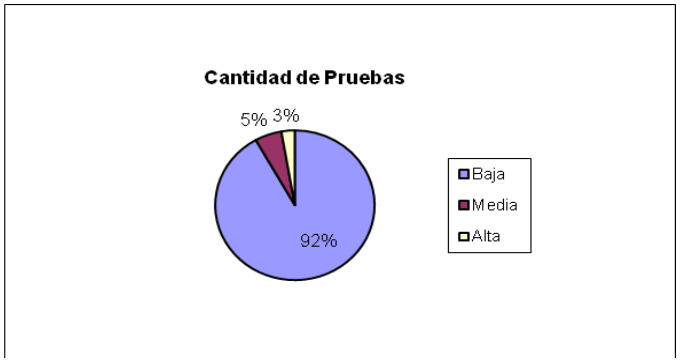
Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.



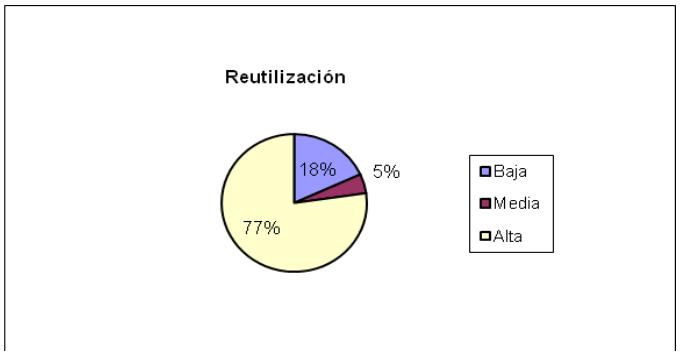
Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.



Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.



Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.



Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del componente Marco de Trabajo tiene una buena calidad pudiéndose observar que el 69% de las clases posee menos de 3 dependencias de otras clases. Además el 19% de las clases no poseen acoplamiento con otras y el 50% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento y Cantidad de Pruebas se comportaron satisfactoriamente para un 92% y el de Reutilización para un 77% de las clases.

4.2 Pruebas de Software

La IEEE define las pruebas como una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, y se observan o almacenan los resultados realizándose una evaluación de algún aspecto del sistema o componente, para validar la calidad y funcionalidad del producto, con el fin de detectar errores, que impidan que dicho producto responda con las necesidades especificadas previamente.

Casos de Prueba

La realización de los casos de pruebas tiene como objetivo demostrar al cliente la reacción que corresponderá por parte del sistema luego de realizar alguna acción en el mismo. Para un mejor entendimiento de las respuestas o posibles funcionalidades que brindará el sistema, según la necesidad del usuario.

Caso de Uso Buscar Persona

1. Condiciones de ejecución

- El sistema debe estar instalado y ejecutándose correctamente.
- El actor debe estar autenticado con los permisos necesarios.

2. Sesiones a probar

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1 Buscar Persona	**EC 1.1** Buscar Persona correctamente	Busca todas las personas asociados al criterio de búsqueda
	EC 1.2 Buscar Persona incorrecta	Al intentar buscar los datos por un criterio de búsqueda incorrecto muestra un mensaje de error.

3. Descripción de variables

No.	Nombre de campo	Clasificación	Valor nulo	Descripción
[1]	Cédula	Componente Cédula	Puede que sea nulo o no	Primero letras y después números.
[2]	Nombre	Campo de texto	Puede que sea nulo o no	Letras
[3]	Primer Apellido	Campo de texto	Puede que sea nulo o no	Letras
[4]	Segundo Apellido	Campo de texto	Puede que sea nulo o no	Letras

4. Matriz de Datos

Escenario	Cédula	Nombre	Primer Apellido	Segundo Apellido	Respuesta del sistema	Resultado de la prueba	Flujo central

EC 1.1 Buscar Persona	V V123456	V Jorge	V Rueda	V García	El sistema realiza la búsqueda de una persona determinada y lo mostrará.	Satisfactoria	Buscar Persona
	V V123456	NA	NA	NA	El sistema realiza la búsqueda de una persona determinada y lo mostrará.	Satisfactoria	
	NA	V Jorge	NA	NA	El sistema realiza la búsqueda de una persona determinada y lo mostrará.	Satisfactoria	
	NA	NA	V Rueda	NA	El sistema realiza la búsqueda de una persona determinada y lo mostrará.	Satisfactorio	
	NA	NA	NA	V García	El sistema realiza la búsqueda de una persona determinada y lo mostrará.	Satisfactoria	
EC 1.2	Incorrect	NA	NA	NA	El sistema deberá		

Buscar Persona incorrecta	o				mostrar un mensaje con los datos del error.	Satisfactoria	
	NA	Incorrecto	NA	NA	El sistema deberá mostrar un mensaje con los datos del error.	Satisfactoria	
	NA	NA	Incorrecto	NA	El sistema deberá mostrar un mensaje con los datos del error.	Satisfactoria	
	NA	NA	NA	Incorrecto	El sistema deberá mostrar un mensaje con los datos del error.	Satisfactoria	

4.3 Conclusiones

En este capítulo se analizaron los casos de prueba y las métricas pertinentes que se le aplicaron al componente para verificar los niveles de calidad con que el mismo fue realizado. Los resultados obtenidos fueron satisfactorios luego de haber aplicado estos métodos. Aunque la mejor prueba para la validación es que el software ha cumplido con todas las funcionalidades requeridas y se encuentra en explotación dentro del proyecto desde hace 3 meses. El mismo fue avalado recientemente por el proyecto de calidad de la facultad con resultados satisfactorios, y actualmente se encuentra listo para ser certificado por Calisoft.

Conclusiones Generales

El resultado de esta investigación fue la obtención de un Marco de Trabajo que garantiza la integración entre los subsistemas Centro de Digitalización y Sistema de Gestión de Antecedentes Penales. Para lograrlo se realizó un estudio de las tendencias y tecnologías actuales más utilizadas, a fines con la temática que se aborda, seleccionando y fundamentando aquellas que más se adecuaban al entorno donde se iban a aplicar, teniendo en cuenta siempre que fueran, en su mayoría, herramientas de código abierto y libres de cualquier tipo de restricción.

La solución fue probada mediante pruebas de caja negra a cada uno de los diferentes módulos donde se verificó que se cumplieran todos los requisitos funcionales. Además se realizó una validación del diseño mediante la utilización de instrumentos de medición que se inspiraron en métricas para la calidad del diseño. Los resultados arrojados permitieron concluir que el diseño presenta valores positivos en indicadores de calidad tales como Reutilización, Facilidad de Mantenimiento, Complejidad del Diseño, Complejidad de Implementación, Acoplamiento, Cantidad de pruebas, entre otros; todo esto unido a la certificación del proyecto de calidad de la facultad, apoya la afirmación de que el diseño e implementación del Marco de Trabajo tiene una muy buena calidad.

Recomendaciones

Al terminar el siguiente trabajo se proponen las siguientes recomendaciones:

- Insertar al Marco de Trabajo un módulo para el desarrollo de flujo de trabajos que ayude y permita controlar los diferentes flujos dentro de las aplicaciones desarrolladas.
- Mejorar los componentes desarrollados para la interfaz de usuario a partir del uso de nuevas tecnologías como swingx y otras existentes.

Bibliografía

1. Oracle Corporation**.** NetBeans. NetBeans IDE 6.9 Release Information*.* [En línea] Oracle Corporation, 2010. [Citado el: 12 de febrero de 2011].
   Disponible en: http://netbeans.org/community/releases/67/.
2. Eclipse, Inc**.** Eclipse Documentation - Current Release (Eclipse Galileo). Help Eclipse SDK. [En línea] [Citado el: 13 de febrero de 2011]. Disponible en: http://help.eclipse.org/galileo/index.jsp
3. Oracle Corporation**.** Oracle Sun Developer Network (SDN). GlassFish Community*.* [En línea] Oracle Corporation. [Citado el: 20 de febrero de 2011]. Disponible en: https://glassfish.dev.java.net/.
4. Visual Paradigm Organización. 2009. Sitio Web oficial Visual-Paradigm. Sitio Web oficial Visual-Paradigm. [En línea] [Citado el: 25 de Febrero 2011]. Disponible en: http://www.visual-paradigm.com/product/vpuml/.
5. PostgreSQL. 2008. PostgreSQL. PostgreSQL. [En línea] PostgreSQL, 2008. [Citado el: 17 de enero de 2011.]. Disponible en: http://www.postgresql.org/about/.
6. Larman C .UML y Patrones: “Introducción al análisis y diseño de la programación Orientada a Objetos”. [En línea] [Citado el: 23 de marzo 2011]. Disponible en: http://bibliodoc.uci.cu/pdf/reg00061.pdf.
7. Pressman, Roger S. 1997. Ingeniería de Software un enfoque práctico. Quinta Edición. . [En línea] [Citado el: 24 de abril 2011].
8. Pressman. 2001. Ingeniería del software Un enfoque práctico. s.l.: Mcgraw-hill. [En línea] [Citado el: 29 de abril 2011].
9. Figueroa, Roberth G. y Solis, Camilo J.2007.Metodologias Tradicionales Vs. Metodologías Ágiles. [En línea] [Citado el 5 de enero 2011].
10. Jacobson, Ivar, Rumbaugh James y Booch, Grady. 2007. El lenguaje Unificado de Modelado. Manual de Referencia. Madrid: Addison Wesley, 2007. Segunda Edición. [En línea] [Citado el: 18 de enero 2011].
11. Menéndez, Rosa. Proyectos [En línea] [Citado el: 3 de marzo 2011]. Disponibles en: http://www.usmp.edu.pe/publicaciones/boletin/fia/info36/proyectos.html#a.

BIBLIOGRAFIA

12. Somerville, Ian. 2005. Ingeniería de Software - 7ma Edición. Madrid: Addison Wesley, 2005. ISBN-84-7829-074-5. [En línea] [Citado el: 10 de marzo 2011].
13. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. El proceso Unificado de Desarrollo de Software. Madrid: Addison Wesley, 2000. 84-7829-036-2. [En línea] [Citado el: 15 de marzo 2011].
14. Conferencia 7 Prueba. [En línea] [Citado el: 4 de mayo 2011]. Disponible en: http://eva.uci.cu.
15. Ingeniería de Software I Conferencia #5Flujo de Trabajo Implementación. [En línea] [Citado el: 21 de abril 2011]. Disponible en: http://eva.uci.cu.
16. Colectivo de autores, 2010, Consultas en JPA-QL.
17. Alexis Moussine-Pouchkine GlassFish Team Sun Microsystems, GlassFish v2.1 clustering.
18. Mike Keith and Merrick Schincariol, Pro JPA 2.
19. Robert Eckstein, Mar Loy and Dave Wood, Java Swing.
20. Colectivo de Autores, 2011, Un vistazo a Swing.
21. Andrew Lee Rubinger and Bill Burke, 2010, Enterprise JavaBeans 3.1.
22. Jim Waldo, Java: The Good Parts.
23. Dhanji R. Prasanna, Dependency Injection.
24. Kathy Sierra and Bert Bates, 2003, Head First EJB.
25. Masoud Kalali, 2010, GlassFish Security.
26. Colectivo de autores, 2010, The Java EE 6Tutorial
27. Jürgen Petri, 2010, NetBeans Platform 6.9 Developer's Guide.
28. Scott Lao. Asp.Net MVC. [En línea][Citado el: 4 de enero 2011]. Disponible en: http://thinkingindotnet.wordpress.com/2007/11/18/aspnet-mvc-framework-primera-parte/
29. Framework Dalas.[En línea] [Citado el: 8 de enero de 2011].Disponible en: http://www.ecured.cu/index.php/Framework_Dalas
30. Juergen Hoeller. Introducción a Spring Framework. [En línea] [Citado el: 21 de enero de 2011]. Disponible en: http://www.palermo.edu/ingenieria/downloads/introduccion_spring_framework_v1.0.pdf
31. Ernesto Bacón Pantoja. El patrón de diseño Modelo-Vista-Controlador y su implementación en Java Swing. [En línea] [Citado el: 12 de enero de 2011]. Disponible en: http://www.ucbcba.edu.bo/Publicaciones/revistas/actanova/documentos/v2n4/v2.n4.bascon.pdf

32. José Enrique González Cornejo. Arquitectura en Capas. [En línea] [Citado el: 12 de enero de 2011]. Disponible en: http://www.docirs.cl/Arquitectura_Tres_Capas.htm
33. José Valle. Arquitectura Cliente-Servidor. [En línea] [Citado el: 13 de enero de 2011]. Disponible en: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.
34. Marcelo Romos. El Sistema Operativo Windows. [En línea] [Citado el: 15 de enero de 2011]. Disponible en: http://publiespe.espe.edu.ec/librosvirtuales/informatica-basica/informatica-basica/informatica-basica02.pdf
35. Anders Helsberg. Introducción al C Sharp. [En línea] [Citado el: 14 de enero de 2011]. Disponible en: http://www.csharp-station.com/Tutorial.aspx.
36. Oscar Belmonte Fernández. Introducción al lenguaje de programación Java. [En línea] [Citado el: 24 de enero de 2011]. Disponible en: http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf
37. Erix Bert. Linux. [En línea] [Citado el: 18 de enero de 2011]. Disponible en: http://www.monografias.com/trabajos/solinux/solinux.shtml.

Anexo1. Descripción de las clases

Nombre	GestorTrazaSistema
Tipo de Clase:	Controladora.
Descripción	
Clase que se encarga de gestionar las trazas del sistema	
Métodos	**Descripción**
insertarTrazaSistema()	Inserta una traza en la base datos.

Nombre	GestorMarcoTrabajo
Tipo de Clase:	Controladora.
Descripción	
Clase encargada de gestionar el flujo de acciones.	
Métodos	**Descripción**
adicionarEventosBotones(final AccionPanel acción)	Configurar los listener y los adiciona para la nueva acción.
mostrarFormulario(IAccion acción)	Método por el cual se inicia.
iniciarNuevoAsistente(AccionPanel acción)	Verifica si hay un flujo en curso, en cuyo caso solicita confirmación del usuario si desea cancelar el actual.
mostrarPanelEnFlujo(AccionPanel acción)	Se guarda la clave de la acción anterior para saber dónde ir cuando se presione el botón btnAnterior y se guarda una referencia de la acción que se va a ejecutar en cada momento.
mostrarPanelEnFlujo(String nombreAccion)	Muestra el panel en flujo dado una el nombre de una acción como parámetro.
crearNuevoPanel(AccionPanel acción)	Crea un nuevo panel.

getPanelExistente(String clave)	Método que busca y retorna una acción por su clave correspondiente.
mostrarFormularioExistente(AccionPanel acción)	Método que muestra la acción pasada como parámetro
mostrarFormularioExistente(String nombreAccion)	Método que muestra una acción dada su clave.
finalizarFlujo()	Finaliza el flujo actual, limpiando el historial de acciones, reiniciando el asistente y el área de trabajo
getAsistente()	Retorna el Wizard.
reiniciarFlujo()	Reinicia el flujo.
obtenerAccionPrincipal()	Obtiene la acción principal.
getAccionActiva()	Retorna la acción actual.

Nombre	UIUtil
Tipo de Clase:	Controladora.
Descripción	
Gestiona los colores y el fondo del panel.	
Métodos	**Descripción**
getColor(String etiqueta)	Retorna el color de la etiqueta.
cambiarFondoPanel(JComponent componente, Color color)	Cambia el fondo del panel.

Nombre	GestorImpresion

Tipo de Clase:	Controladora.
Descripción	
Gestiona la impresión de documentos.	
Métodos	**Descripción**
getEstadoImpresion()	Obtiene el estado de impresión.
imprimir(JasperPrint documento, boolean seleccionarImpresora)	Imprime un documento.
imprimir(JasperPrint documento, int paginaInicial, int paginaFinal, boolean seleccionarImpresora)	Imprime un documento pasándole como parámetro la pagina inicial y final de lo que se desea imprimir.

Nombre	I18n
Tipo de Clase:	Controladora.
Descripción	
Gestiona los mensajes y las etiquetas para internacionalización del idioma.	
Métodos	**Descripción**
getEtiqueta(String idEtiqueta)	Retorna la etiqueta.
String getMensaje(String clave)	Retorna un mensaje dado una clave.
getIcono(String clave)	Retorna un icono dado una clave.
getImagen(String clave)	Retorna una imagen dado una clave.
getIdiomaActual()	Retorna el idioma.

getMensajeExcepcion(String clave)	Retorna un mensaje de excepción dado una clave.

Nombre	AccionBase
Tipo de Clase:	Controladora
Descripción	
Clase que define las acciones básicas de los formularios.	
Métodos	**Descripción**
esperaCursor()	Espera la posición del cursor.
defaultCursor()	Pone el cursor en una posición por defecto.
getTitulo()	Obtiene el titulo.
setTitulo(String titulo)	Pone el titulo que pasan por parámetro.
onCerrandoFormulario()	Cierra un formulario.
onMostrandoFormulario()	Muestra un formulario.

Nombre	AccionDialogo
Tipo de Clase:	Controladora
Descripción	
Clase que se encarga de mostrar los mensajes que se le mostraran al usuario	
Métodos	**Descripción**

esConTitulo()	Retorna si tiene titulo o no.
setConTitulo(boolean conTitulo)	Crea una barra de titulo.
getLbltitulo()	Obtiene el titulo.
setLbltitulo(JLabel lbltitulo)	Crea un titulo.
getPnlBarraTitulo()	Obtiene una barra de titulo.
setPnlBarraTitulo(JPanel pnlBarraTitulo)	Pone en el panel la barra de titulo.
cerrar()	Cierra el formulario.

Nombre	AccionPanel
Tipo de Clase:	Controladora
Descripción	
Clase que encapsula todas la funcionalidades que van a tener los paneles.	
Métodos	**Descripción**
setVisibilidadBotones(boolean anteriorVisible, boolean siguienteVisible, boolean terminarVisible, boolean cancelarVisible)	Pone visible los botones anterior, siguiente, terminar y cancelar.
isBtnAnteriorVisible()	Retorna si el botón anterior es visible o no.
isBtnCancelarVisible()	Retorna si el botón Cancelar es visible o no.
isBtnSiguienteVisible()	Retorna si el botón Siguiente es visible o no.

isBtnTerminarVisible()	Retorna si el botón Terminar es visible o no.
isBtnEditarVisible()	Retorna si el botón Editar es visible o no.
getBtnAnteriorTexto()	Obtiene el texto del botón Anterior.
getBtnCancelarTexto()	Obtiene el texto del botón Cancelar.
getBtnEditarTexto()	Obtiene el texto del botón Editar.
getBtnSiguienteTexto()	Obtiene el texto del botón Siguiente.
getBtnTerminarTexto()	Obtiene el texto del botón Terminar.
getBtnAnteriorIcono()	Obtiene el icono del botón Anterior.
setBtnAnteriorIcono(ImageIcon btnAnteriorIcono)	Le inserta un icono al botón Anterior el cual es pasado por parámetro.
getBtnCancelarIcono()	Obtiene el icono del botón Cancelar.
setBtnCancelarIcono(ImageIcon btnCancelarIcono)	Le inserta un icono al botón Cancelar el cual es pasado por parámetro.
getBtnEditarIcono()	Obtiene el icono del botón Editar.
setBtnEditarIcono(ImageIcon btnEditarIcono)	Le inserta un icono al botón Editar el cual es pasado por parámetro.
getBtnSiguienteIcono()	Obtiene el icono del botón Siguiente.
setBtnSiguienteIcono(ImageIcon btnSiguienteIcono)	Le inserta un icono al botón Siguiente el cual es pasado por parámetro.
getBtnTerminarIcono()	Obtiene el icono del botón Terminar.
setBtnTerminarIcono(ImageIcon btnTerminarIcono)	Le inserta un icono al botón Terminar el cual es pasado por parámetro.

Nombre	IAccion
Tipo de Clase:	Interface
Descripción	
Clase que encapsula todas las funcionalidades de las acciones.	
Métodos	**Descripción**
esperaCursor()	Espera la posición del cursor.
defaultCursor()	Pone el cursor en una posición por defecto.
getTitulo()	Obtiene el titulo.
setTitulo(String titulo)	Pone el titulo que pasan por parámetro.
onCerrandoFormulario()	Cierra un formulario.
onMostrandoFormulario()	Muestra un formulario.
inicializarFormulario()	Inicializa los formularios.

Nombre	frmPrincipal
Tipo de Clase:	Interfaz.
Descripción	
Clase que representa el formulario *gestionarPrincipal* en el que se muestran y se trabajan con los datos necesarios para diseñar el formulario Principal.	
Atributos	**Tipo**
lblUsuario	JLabel
lblImagen	JLabel

lblNombreUsuario	JLabel
lkbAyuda	JLinkButton
lkbBloquearSesion	JLinkButton
lkbCambiarContrasena	JLinkButton
lkbCerrarSesion	JLinkButton
panelDeDegradado1	PanelDeDegradado
pnlAreaTrabajo	JPanel
pnlBanner	JPanel
pnlContenedor	JPanel
pnlContenedorMenu	JTaskPane
pnlMenuIzquierdo	JPanel
pnlSubBanner	JPanel
jScrollPane2	JScrollPane

Nombre	AccionPrincipal
Tipo de Clase:	Controladora
Descripción	
Clase acción que controla las funcionalidades del formulario Principal.	
Métodos	**Descripción**

ejecutar()	Carga los recursos del componente Splash
cerrar()	Cierra la ventana principal
iniciarFlujo()	Carga los recursos y los atributos a mostrar en la ventana principal.
onCambiarContrasena()	Invoca la acción AccionCambiarContrasena y muestra el formulario frmCambiarContrasena para cambiar la contraseña.
onCerrarSesion()	Cierra la sesión que se encuentra activa.
cargarBannereIcono()	Carga el icono que se muestra en el panel.
configurarSesionUsuario()	A partir de la sesión autenticada carga en el menú las acciones del rol base.
generarMenu()	Llena el menú con los distintos módulos en dependencia de la sesión inicializada.
intercambiarTagsMenu()	Garantiza que solo haya un módulo abierto a la vez en la ventana principal.
llenarListaModulos()	Llena los módulos con las acciones asociadas a ellos.
limpiarAreaTrabajo()	Limpia y actualiza el área de trabajo
internacionalizacion()	Crea un atributo del tipo frmPrincipal y añade las etiquetas del paquete i18N.

Nombre	GestorNCSesiones
Tipo de Clase:	Controladora
Descripción	
Clase que gestiona la Lógica de Negocio del Cliente para la entidad Sesión. Realiza las llamadas correspondientes a la Lógica de Negocio del Servidor para completar las funcionalidades.	

Métodos	Descripción
buscarUsuarioSesion()	Invoca a la fachadaGestoresRemotos para conectarse con la lógica del negocio del servidor.
deshabilitar()	Deshabilita la sesión que es pasada por parámetro.
adicionarSesionActiva()	Conecta con la lógica de negocio de la parte servidor y crea una sesión que es pasada por parámetro.
buscarSesionPorIdUsuario()	Busca una sesión dado un id de un usuario.
buscarSesion()	Busca la sesión a la que pertenece el usuario dado un nombre y el primer apellido.
eliminarSesionActiva()	Elimina una sesión dado un id de un usuario. Este id debe pertenecer a un usuario que se encuentre actualmente activo en el sistema.
configurarSesionUsuario()	A partir de la sesión autenticada carga en el menú las acciones del rol base.
generarMenu()	Llena el menú con los distintos módulos en dependencia de la sesión inicializada.
intercambiarTagsMenu()	Garantiza que solo haya un módulo abierto a la vez en la ventana principal.
llenarListaModulos()	Llena los módulos con las acciones asociadas a ellos.
limpiarAreaTrabajo()	Limpia y actualiza el área de trabajo
internacionalizacion()	Crea un atributo del tipo frmPrincipal y añade las etiquetas del paquete i18N.

Nombre	IconoCellRenderer
Tipo de Clase	Controladora.

Descripción	
Clase que permite poner un icono en las celdas de un JTable.	
Métodos	**Descripción**
getTableCellRendererComponent()	Método que redefine las celdas de la tabla.
fillColor()	Este método es para que pinte el fondo del JLabel cuando lo seleccionamos para que no quede en blanco, desentonando con el resto de las celdas que no son JLabel.

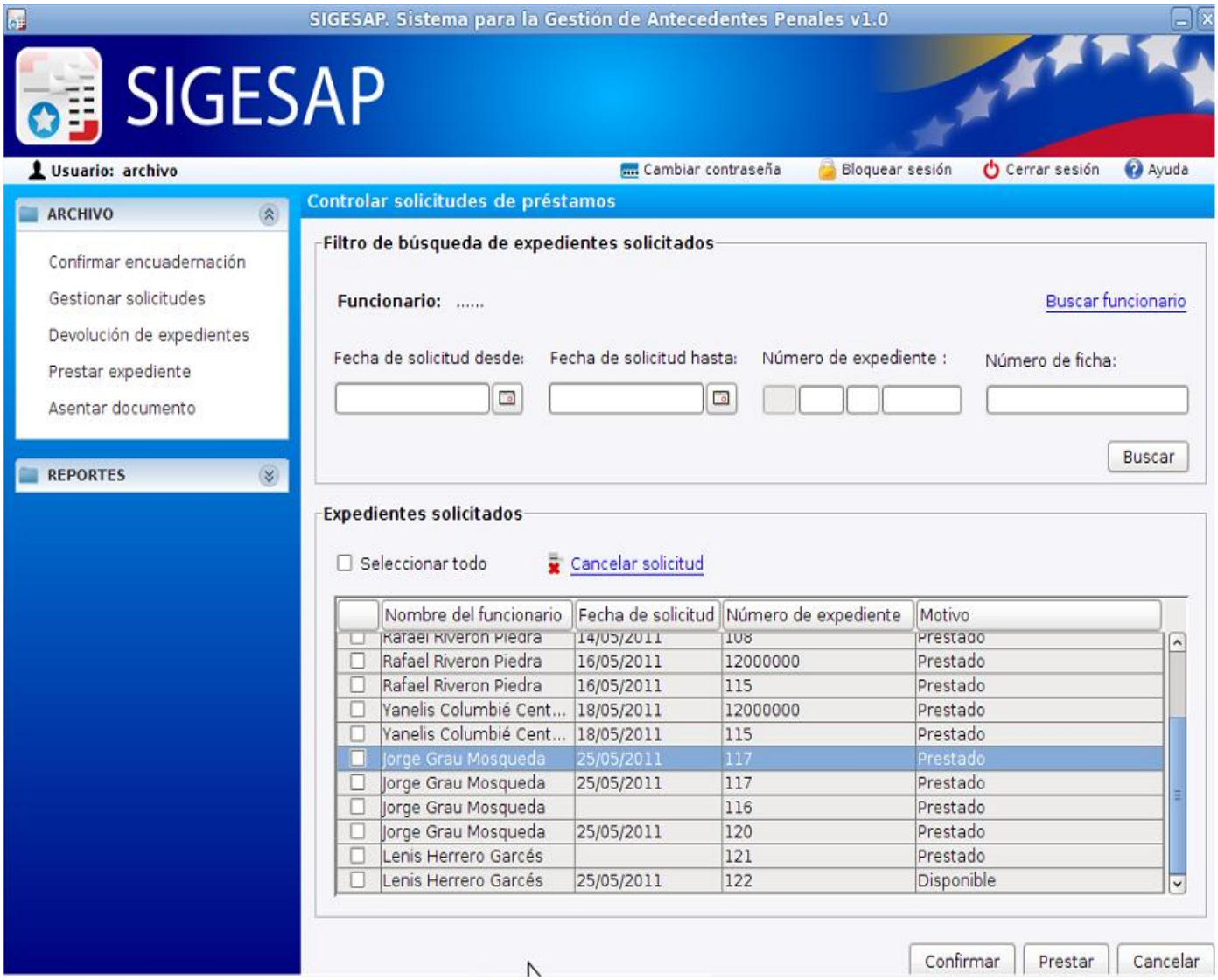
Nombre	AccionMenu
Tipo de Clase	Controladora.
Descripción	
Clase encargada crear el menú.	
Métodos	**Descripción**
getAccion()	Método devuelve una IAccion.

Nombre	Wizard
Tipo de Clase	Controladora.
Descripción	

Clase encargada gestionar todos los componentes del área de trabajo	
Métodos	**Descripción**
adicionarAccion()	Dada una acción, la adiciona y la muestra.
setVisibilidadBotones()	Muestra los botones correspondientes al área de trabajo.
setTextoBotones()	Muestra el texto de los botones a mostrar.
setIconoBotones()	Muestra el icono
ponerColor()	Da color al fondo del área de trabajo.

Nombre	BarraHerramientasFlujo
Tipo de Clase	Controladora.
Descripción	
Clase que hereda de JToolBar y centraliza el trabajo.	
Métodos	**Descripción**
adicionarComponentes()	Adiciona un componente a la barra de herramienta.
instalarListeners()	Elimina un componente de la barra de herramienta.
instalarCombinacionesTeclas ()	Permite utilizar el teclado para realizar operaciones sobre el área de trabajo.
ListenerBarraHerramientasFlujo ()	Muestra en la barra de herramientas todas las operaciones que se pueden realizar.

Anexo2. Imágenes



Aplicación en ejecución.

Edit Data - serv233admin (10.32.18.233:5432) - sigesap.pruebas - framework.t_traza

Archivo Editar Vista Herramientas Ayuda

Sin límite

	id_traza [PK] double precision	id_usuario double precision	fecha_traza timestamp without time zone	descripcion_traza character varying(500)	ip character varying(255)	ip_double double precision	sistema boolean
1	101.0001	101.46	2011-05-27 02:50:16.921	Sesion finalizada:	10.32.18.121	10032018121	TRUE
2	101.0011	101.38	2011-05-27 08:38:43.206	Procesado	192.168.177.220	192168177220	FALSE
3	101.0012	101.44	2011-05-27 17:37:43.53	Sesion finalizada:	10.32.18.101	10032018101	TRUE
4	101.003	101.39	2011-05-23 10:29:16.288	Sesion iniciada:	10.32.18.101	10032018101	FALSE
5	101.004	101.34	2011-05-24 08:53:49.457	Cesión cerrada.	10.8.30.139	10008030139	TRUE
6	101.005	101.33	2011-05-25 03:14:55.699	Sesion iniciada:	10.32.18.157	10032018157	TRUE
7	101.006	101.34	2011-05-24 19:41:20.941	Sesion finalizada:	10.32.18.101	10032018101	TRUE
8	101.007	101.35	2011-05-25 21:26:02.906	Sesion iniciada:	10.32.18.153	10032018153	TRUE
9	101.008	101.39	2011-05-26 22:58:44.444	Sesion finalizada:	192.168.177.217	192168177217	TRUE
10	101.009	101.38	2011-05-26 11:42:04.546	Sesion iniciada:	10.32.18.22	10032018022	TRUE
11	101.0101	101.04	2011-05-26 15:55:46.061	Sesion finalizada:	10.32.18.22	10032018022	TRUE
12	101.0102	101.44	2011-05-26 16:02:48.559	Sesion finalizada:	10.32.18.101	10032018101	TRUE
13	101.0103	101.34	2011-05-27 03:21:04.703	Sesion finalizada:	10.32.18.121	10032018121	TRUE
14	101.0104	101.44	2011-05-26 16:20:01.913	Sesion finalizada:	10.32.18.101	10032018101	TRUE
15	101.0106	101.39	2011-05-27 03:59:48.327	Cesión cerrada.	10.32.18.121	10032018121	TRUE
16	101.0107	101.44	2011-05-26 18:22:39.802	Sesion iniciada:	10.32.18.101	10032018101	TRUE
17	101.0108	101.44	2011-05-26 20:11:09.545	Sesion iniciada:	10.32.18.101	10032018101	TRUE
18	101.0109	101.34	2011-05-26 23:32:12.996	Creado	192.168.177.220	192168177220	FALSE
19	101.0111	101.34	2011-05-27 09:29:36.502	Sesion finalizada:	192.168.177.220	192168177220	TRUE
20	101.0112	101.34	2011-05-27 12:18:03.364	Sesion finalizada:	192.168.177.220	192168177220	TRUE
21	101.0113	101.46	2011-05-27 15:30:59.523	Sesion finalizada:	192.168.177.222	192168177222	TRUE
22	101.0114	101.33	2011-05-27 15:34:13.071	Sesion iniciada:	10.32.18.101	10032018101	TRUE
23	101.0115	101.33	2011-05-27 15:48:57.452	Cesión cerrada.	10.32.18.101	10032018101	TRUE
24	101.0116	101.33	2011-05-27 15:52:09.393	Sesion finalizada:	10.32.18.101	10032018101	TRUE
25	101.0117	101.33	2011-05-27 16:17:53.225	Sesion iniciada:	10.32.18.101	10032018101	TRUE
26	101.0118	101.33	2011-05-27 16:32:37.479	Sesion iniciada:	10.32.18.101	10032018101	TRUE
27	101.0119	101.44	2011-05-27 16:58:09.28	Sesion finalizada:	10.32.18.101	10032018101	TRUE
28	101.0121	101.44	2011-05-30 00:44:23.548	Sesion finalizada:	10.32.18.101	10032018101	TRUE
29	101.0122	101.44	2011-05-30 01:27:48.382	Sesion finalizada:	10.32.18.101	10032018101	TRUE
30	101.0123	101.43	2011-05-30 02:37:05.06	Sesion finalizada:	10.32.18.101	10032018101	TRUE
31	101.0124	101.44	2011-05-30 08:12:52.862	Sesion finalizada:	10.32.18.101	10032018101	TRUE
32	101.0125	101.04	2011-05-30 08:47:10.202	Cesión cerrada.	10.32.18.101	10032018101	TRUE
33	101.0126	101.34	2011-05-30 08:51:21.153	Sesion iniciada:	10.32.18.101	10032018101	TRUE
34	101.0127	101.33	2011-05-30 20:38:07.738	Sesion iniciada:	10.32.18.116	10032018116	TRUE
35	101.029	101.39	2011-05-23 09:31:52.053	Sesion iniciada:	10.32.18.101	10032018101	FALSE

932 rows.

Trazas almacenadas en la Base de Datos