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

Título:Sistema de gestión de datos para las solicitudes de trámites no presenciales de pasaportes en Cuba.

Autores:

Yoelmy Hernández Barzaga

Ariel Leandro Doimeadios Aguilera

Tutores:

MsC. Adrian Alberto Machado Cento

Ing. Reynier Blanco Zambrano

La Habana, Junio de 2014

"Año 56 de la Revolución"

"Las cadenas de la esclavitud solamente atan las manos:
es la mente lo que hace al hombre libre o esclavo"

Franz Grillparzer

Declaración de autoría

Se declara que somos los únicos autores del trabajo titulado: “Sistema de gestión de datos para la solicitud de trámites no presenciales de pasaportes en Cuba”, y se otorga a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los __ días del mes de _____ del año ____

Yoelmy Hernández Barzaga

_____

Firma del autor

Ariel Leandro Doimeadios Aguilera

_____

Firma del autor

MsC. Adrian Alberto Machado Cento

_____

Firma del tutor

Ing. Reynier Blanco Zambrano

_____

Firma del tutor

Datos de contacto

Tutor: MsC. Adrian Alberto Machado Cento

Ingeniero en Ciencias Informáticas, UCI 2006.

Categoría docente: Profesor Instructor.

Graduado de Ingeniería en Ciencias Informáticas en la UCI en el año 2006, Máster en Informática Aplicada en la UCI en el año 2008, Profesor del departamento de Desarrollo del Centro de Identificación y Seguridad Digital. Jefe del proyecto Identidad Cuba

Correo electrónico: amachado@uci.cu

Tutor: Ing. Reynier Blanco Zambrano

Ingeniero en Ciencias Informáticas, graduado en el 2009. Cuenta con varios años de experiencia en el proyecto Identidad Cuba donde se ha desempeñado como Arquitecto Principal y actualmente Jefe de Desarrollo del Sistema Único de Identificación Nacional (SUIN).

Correo electrónico: rblanco@uci.cu

Dedicatoria

Especialmente a mi mamá, que ha sido y será siempre, mi luz y guía en todo momento, la esencia de lo que soy y seré siempre.

A mi papá, que aunque se encuentra lejos y no está aquí conmigo, sé que está orgulloso de mi y de todos mis logros.

A mi hermano, que aunque no lo demuestre, sé que también está orgulloso de mis logros.

A mi abuela Oneida, por ser la que ha permitido este momento y porque sé que nunca ha dejado de creer.

A mi familia por su apoyo y preocupación.

A mis amigos de la universidad, sin ellos este camino hubiera sido más difícil.

Yoelmy

A mis padres que han sido los pilares de mi desarrollo y el sustento con que siempre he contado para llegar hasta aquí.

A mi novia Lilien que con su amor y apoyo he logrado muchas cosas desde que nos conocimos, esta es una de ella.

A toda mi familia en general por sus aportes a este resultado.

A todos mis amigos, dentro y fuera de la universidad.

Ariel Leandro

Agradecimientos

A mi mamá, por todo su apoyo, consejos y comprensión en estos años de universidad. Gracias mamá por estar siempre presente, atenta y dispuesta a ayudarme en todo momento; sin ti nada de esto fuera posible. Recuerda siempre que este título es más tuyo que mío porque siempre has sido y serás: la gran arquitecta de todos estos resultados. Gracias una y mil veces por ser cómo eres, por inspirar ese amor y dedicación que siempre transmites y sobre todo por nunca dejarte vencer por los obstáculos y enseñarme que todo es posible.

A mi papá, por su apoyo y comprensión. Gracias flaco por estar siempre atento y preocupado por mí, por tenerme siempre presente y por tu ayuda.

A mi hermano, por su preocupación y su apoyo. Gracias brother, porque aunque no lo demuestras sé que también estas orgulloso de este momento.

A mi abuela Oneida, por permitir este momento, por su apoyo y preocupación y por tener siempre un consejo que brindar y nunca dejar de orar, aún en los momentos más difíciles.

A mi familia, por estar siempre atentos y preocupados.

A mis amigos de la universidad, de ellos he aprendido mucho en estos años y han sido de gran ayuda en cualquier momento.

Yoelmy

A mis padres por todo el amor que me han ofrecido en toda mi vida para lograr nuestro sueño de ser profesional:

A Caridad (Cachi) mi madre por estar ahí siempre que la he necesitado dándome el apoyo necesario como toda buena madre lo hace con su hijo, gracias mamá te amo.

A Ariel (Japo) mi padre por ser un amigo incondicional y el mejor ejemplo a seguir mi ídolo en todo, siempre me dices que el escalón más grande de la especie humana es ser un hombre digno y honrado, gracias papá por ayudarme en mi camino por la vida.

A la persona que me está acompañando por esta etapa de mi vida mi novia Lilien la cual me ha brindado todo su amor sin pedir nada a cambio, gracias nena te amo.

A mis hermanos por su ayuda en toda mi vida, los quiero mucho.

A mi familia en general por todas las contribuciones que tuvieron en mi desarrollo.

A todos mi amistades dentro y fuera de la universidad.

A todas las personas que de una forma u otra ayudaron en mi formación y desarrollo para ser la persona que soy.

Ariel Leandro

Resumen

Los trámites de pasaportes solicitados por las entidades estatales en Cuba son realizados por el Ministerio del Interior o el Ministerio de Relaciones Exteriores. Por este motivo cualquier trabajador perteneciente al sector estatal que necesite viajar al exterior por asuntos laborales se le tramita la documentación a través de la Dirección de Identificación, Inmigración y Extranjería del Ministerio del Interior. Los datos que conforman estas solicitudes quedan registrados en documentos con formato duro, esta información se traslada sin estar cifrada de algún modo, por ello están expuestos a pérdidas o irregularidades en la información y el proceso de gestión es lento. El presente trabajo persigue desarrollar un sistema informático capaz de garantizar la gestión de datos para la tramitación no presencial de pasaportes que cumplan con los estándares internacionales en materia de documentos de viaje. Para lograr esta propuesta de solución se realizó un estudio de soluciones existentes para trámites de pasaportes; se logró desarrollar una aplicación de escritorio con ejecución multiplataforma sobre el entorno de desarrollo *Netbeans* 7.4, utilizando el lenguaje *Java*, con PostgreSQL 9.2.4 como gestor de base de datos, haciendo uso del *framework Hibernate* 3.6.10 para el mapeo objeto-relacional y teniendo como guía la metodología de desarrollo *eXtreme Programming*. La eficiencia y funcionalidad de la aplicación se valida mediante la ejecución de casos de prueba para verificar el correcto funcionamiento. El despliegue de la propuesta logrará aportar seguridad en el traslado y manejo de la información, utilizando criptografía mixta, y disminuirá los tiempos del proceso de trámite.

Palabras claves: aplicación de escritorio, cifrado, multiplataforma, trámites de pasaporte.

Índice de contenido

Índice de Ilustraciones

Índice de Tablas

Introducción

Producto a las grandes olas migratorias provocadas por los conflictos bélicos internacionales del siglo pasado, aumenta la necesidad de establecer normas de control más estrictas en los trámites de viajes, fenómeno que trajo consigo la aparición de nuevos parámetros de seguridad en la confección del pasaporte. El pasaporte es de carácter oficial y personal, posee validez internacional, y es expedido por las entidades autorizadas en cada país, permitiendo el autorizo legal para el ingreso y la salida del mismo (1).

Los trámites de pasaportes existen desde el siglo XX, llevando a las sociedades a una constante adaptación en cuanto a normas y estándares en materia de viajes. El permanente crecimiento de las necesidades sociales y el amplio uso de los documentos de viaje han propiciado que los países lograran ir perfeccionando las vías de identificación personal para estos trámites. En 2010 unos doscientos catorce millones de personas, lo que representa el tres por ciento de la población mundial, no vive en su país de origen, según datos estadísticos publicados por el Fondo de Población de las Naciones Unidas (UNFPA[1] por sus siglas en inglés) (2).

Desde sus inicios los trámites de pasaporte han traído consigo largos métodos para terminar con éxito todo el procedimiento de trámite. Las gestiones de este tipo, desde un principio, han sido llevadas a cabo de forma manual por lo que el movimiento y tratamiento de la información no se realiza de una manera segura. Por este motivo la existencia de errores e irregularidades en la confección de pasaportes, se presentan con gran frecuencia.

El amplio uso y aceptación de las tecnologías de la información ha permitido que los procesos sean ejecutados de un modo fiable, económico y eficiente. Se logra un notable desarrollo y un menor tiempo de respuesta en los procesos, por lo que el despliegue de estos avances tecnológicos en las organizaciones y sistemas empresariales ha sido vertiginoso. En este sentido, las entidades encargadas de efectuar los trámites de viaje no han quedado rezagadas y en la actualidad la informatización de los procesos de confección de pasaportes, ofrece un incremento del control y la seguridad combinándose con los métodos manuales vigentes.

Según el Decreto-Ley 302 del 2012 se establece que las entidades cubanas designadas para la confección de pasaportes son el Ministerio de Relaciones Exteriores(MINREX), encargado de confeccionar los pasaportes Diplomáticos y de Servicio, y el Ministerio del Interior(MININT) designado para la confección de los pasaportes Oficial, Corriente y de Marino (3). En Cuba, la emisión de pasaporte Corriente cuenta con el Sistema Único de Identificación Nacional que ofrece ventajas a diferencia de los restantes tipos de

[1] *United Nation Fund for Population Activities* (Fondo de Población de las Naciones Unidas).

pasaportes entre ellas el chequeo biométrico a la identidad del ciudadano y la verificación contra la base de datos única del registro de personas. Las ventajas que posee el pasaporte Corriente favorecen la efectividad en la gestión de los trámites de pasaporte, realiza un óptimo registro y control de los ciudadanos (4).

Las personas vinculadas laboralmente al sector estatal, que por razones institucionales precisen viajar al extranjero, hacen la solicitud de pasaporte desde su entidad laboral y la misma se encarga de tramitar la documentación a la Dirección de Identificación, Inmigración y Extranjería del Ministerio del Interior (3). La captura de datos personales para iniciar el proceso de trámites se ve afectado por diversos problemas tales como:

- Los trámites de pasaportes para el caso que no sea pasaporte Corriente no cuentan con un proceso de chequeo biométrico.
- El proceso de confección de las solicitudes de pasaportes efectuado en las entidades estatales o consulados no permite una verificación con la base de datos única del ciudadano.
- El traslado de la información hacia las oficinas expedidoras no es realizado de manera segura, pudiendo existir pérdidas y/o modificación de los datos personales capturados.
- Las solicitudes se registran en formato duro.
- Grandes volúmenes de datos físicos.
- El desarrollo actual del proceso de trámites es lento debido a cómo se trabajan las solicitudes de pasaportes, a todo esto se suma la existencia de una gran cantidad de trabajadores cubanos prestando servicio en el exterior en actividades oficiales; tan solo en el sector de la salud, hasta agosto de 2012, existían alrededor de 31 000 profesionales brindando servicios en 69 países (5).

Dada la **situación problemática** planteada anteriormente se formula el siguiente **problema científico**: ¿Cómo mejorar el proceso de gestión de datos para efectuar trámites de pasaportes de manera no presencial en el sector estatal en Cuba?

Se presenta como **objeto de estudio** el proceso de tramitación de pasaportes.

El **objetivo de la investigación** es desarrollar un sistema informático que permita la gestión de datos para la tramitación no presencial de pasaportes en Cuba, que cumplan con los estándares internacionales en materia de documentos de viaje.

El **campo de acción** lo constituye el proceso de captura y manejo de datos para la tramitación de pasaportes en el sector estatal de Cuba.

Teniendo en cuenta los siguientes **objetivos específicos:**

- Diagnosticar el estado actual de los elementos teóricos asociados a los procesos de trámites de pasaportes.

- Realizar análisis y diseño del sistema de gestión para la tramitación no presencial de pasaportes en Cuba.
- Implementar el sistema de gestión para la tramitación no presencial de pasaportes en Cuba.
- Validar la calidad del software mediante pruebas.

Planteándose la siguiente **idea a defender:** el desarrollo de un sistema informático para el proceso de gestión de datos con los cuales solicitar pasaportes de manera no presencial en el sector estatal de Cuba; logrará aumentar la seguridad en el manejo y traslado de la información para evitar pérdidas y/o modificaciones de los datos, eliminar los grandes volúmenes físicos de solicitudes y mejorar los tiempos de respuesta del proceso.

Los **métodos científicos** que guiaron la investigación fueron:

- **Métodos teóricos**
  - **Analítico-Sintético:** la aplicación de este método se evidencia en el estudio de la documentación existente, asegurando un mejor entendimiento del proceso de trámites de pasaportes y del uso de las herramientas tecnológicas necesarias en el desarrollo. Propicia una mejor elección de los elementos asociados al objeto de estudio.
  - **Histórico-Lógico:** la ejecución del método posibilita analizar el problema, de la tramitación de pasaportes en los distintos sectores que abarca la investigación, en toda su trayectoria histórica. Brinda un acercamiento al proceso de trámites en un periodo de tiempo lógico.
- **Métodos empíricos**
  - **Entrevistas:** la aplicación de este método ayuda a obtener información relacionada con el desarrollo del proceso de trámites de pasaportes en el sector estatal.
  - **Observación:** con la ejecución de este método se puede definir mejor el proceso de trámites y lograr una captura óptima de los requerimientos necesarios para el desarrollo del sistema.

El documento se encuentra divido en tres capítulos:

- **Capítulo 1.Fundametación teórica:** En este capítulo se aborda el estado del arte sobre los trámites de pasaportes a nivel nacional e internacional. El desarrollo tecnológico, metodologías y software usados actualmente en torno al problema dado y en las que se soporta el desarrollo de la solución propuesta. Se realiza un estudio crítico y valorativo donde se refieren los principales conceptos asociados al problema.
- **Capítulo 2. Características, análisis y diseño:** En este capítulo se brinda una descripción de todos los procesos en los que se desarrolla el negocio. Se hace referencia a la información que es manejada, propuesta del sistema, modelo del negocio así como los requerimientos funcionales y no funcionales de la solución. Se relaciona la estructura de los procesos que participan implementando los requerimientos descritos, además de la representación del diseño de clases y base de datos.

- **Capítulo 3. Implementación y prueba:** En este capítulo se desarrolla el modelo de implementación seguido para darle solución a la problemática existente. Queda definida la estructura de los componentes, estándares de programación, tratamiento de errores, modelo de despliegue y los distintos casos de prueba a ejecutar.

Capítulo 1. Fundamentación teórica

En el presente capítulo se aborda el marco teórico de la investigación sobre la captura y manejo de los datos en las solicitudes de trámites de pasaportes en el sector estatal. Se exponen los conceptos claves asociados al dominio del problema en cuanto a la gestión de los datos que se realizan en los trámites de pasaportes. Se elabora un estudio sobre los principales sistemas informáticos de factura nacional e internacional, relacionados con la gestión de datos para trámites de pasaportes. Al mismo tiempo se describen las herramientas, patrones y entornos de desarrollo seleccionados para ejecutar el proceso de modelado y desarrollo de la aplicación.

1.1 Conceptos asociados al dominio del problema.

El pasaporte tuvo sus primeras apariciones hace siglos cuando las personas que necesitaban viajar solicitaban documentos u objetos que les garantizara seguridad en el viaje. Al principio los pasaportes no tenían la forma con la que actualmente se conocen, sino que han ido evolucionando desde disímiles objetos como anillos o un pedazo de papel, denominado salvoconducto, hasta llegar al formato actual con fuertes medidas de seguridad, zonas de lectura mecánica y uso de rasgos biométricos. Las normas que rigen el formato que deben tener todos los pasaportes expedidos fueron establecidas en 1920 por la Liga de las Naciones (antecesor de la Organización de Naciones Unidas). Como parte de su trabajo la Organización de Aviación Civil Internacional (OACI), desde 1986, comenzó su trabajo en las zonas de lectura mecánica cuando se crea el grupo de expertos sobre la tarjeta-pasaporte (1).

1.1.1 Pasaportes.

Según el Decreto-Ley 302 del 2012: “El pasaporte es un certificado de identidad y ciudadanía que expide, según sea el caso, el Ministerio de Relaciones Exteriores o el Ministerio del Interior, para surtir efectos sólo en cuanto a la admisión y estancia de su titular en el extranjero como ciudadano cubano” (3).

En la Ley de Migración de la República de Cuba se establecen cinco tipos de pasaportes a expedir (3):

Pasaporte Diplomático: es un tipo de pasaporte asignado a:

“ a) Miembros del Buró Político y del Secretariado del Comité Central del Partido Comunista de Cuba;

b) Miembros del Comité Central del Partido Comunista de Cuba;

c) Jefes y Vicejefes de Departamentos, Jefes de Secciones, y funcionarios del Comité Central del Partido Comunista de Cuba;

ch) Miembros del Consejo de Estado;

d) Diputados a la Asamblea Nacional del Poder Popular;

e) Integrantes del Consejo de Ministros;

f) Presidentes de Organismos de la Administración Central del Estado que no formen parte del Consejo de Ministros;

g) Secretario General de la Central de Trabajadores de Cuba;

h) Presidente y Vicepresidente del Tribunal Supremo Popular;

i) Fiscal General de la República;

j) Vicefiscales de la Fiscalía General de la República;

k) Jueces del Tribunal Supremo Popular;

l) Primeros Secretarios de los Comités Provinciales del Partido Comunista de Cuba;

ll) Presidentes de las Asambleas Provinciales del Poder Popular;

m) Vicepresidentes y Viceministros de los Organismos de la Administración Central del Estado;

n) Funcionarios Diplomáticos y Consulares de la República, los Consejeros y Agregados Comerciales, Económicos, Culturales, de Prensa, Militares, Aéreos y Navales; los funcionarios del Ministerio de Relaciones Exteriores y los Correos Diplomáticos;

ñ) Jefes de Departamentos de la Asamblea Nacional del Poder Popular, del Consejo de Ministros y su Comité Ejecutivo; Asesores de los Vicepresidentes del Consejo de Estado y del Consejo de Ministros; Asesores y funcionarios del Consejo de Estado, del Consejo de Ministros y de su Comité Ejecutivo, así como de sus respectivas Secretarías;

o) Jefes de Dirección del Ministerio de las Fuerzas Armadas Revolucionarias y del Ministerio del Interior;

p) Jefes de Dirección del Ministerio del Comercio Exterior y la Inversión Extranjera y del Banco Central de Cuba;

q) Delegados a Congresos o Conferencias Internacionales, intergubernamentales o de carácter diplomático, y cuantos otros funcionarios del Partido Comunista de Cuba, del Estado y del Gobierno el Ministro de Relaciones Exteriores estime conveniente y necesario para el cabal cumplimiento de las misiones encomendadas;

r) Los miembros de la familia de las personas relacionadas anteriormente de conformidad con lo previsto en el Artículo número 37 de la Convención de Viena, 1961, y otras Convenciones similares de las que Cuba sea parte.”.

La entidad encargada de expedir este pasaporte es el Ministerio de Relaciones Exteriores. Es válido por un periodo de tres años desde la fecha en que se expide y con posibilidad de prórroga tres veces más por igual cantidad de tiempo. Este pasaporte puede ser conservado por su titular o debe ser entregado a su regreso al país, según sea el motivo de salida.

Pasaporte de Servicio: se otorga este tipo de pasaporte a cualquier persona con cargo administrativo de forma permanente, técnico o auxiliar de las Embajadas, Misiones o Consulados cubanos así como a sus familiares. Este pasaporte expedido por el Ministerio de Relaciones Exteriores, es válido por un periodo de

tres años desde la fecha en que se expide y con posibilidad de prórroga tres veces más por igual cantidad de tiempo.

Pasaporte Oficial: tipo de pasaporte expedido a las personas que no están autorizadas a portar pasaporte Diplomático o de Servicio y necesiten cumplimentar viaje al exterior por asuntos de carácter oficial por interés de su organismo o entidad estatal y a sus familiares. Esta solicitud se realiza en la Dirección de Inmigración y Extranjería. Este pasaporte solo es válido por el tiempo de duración del viaje.

Pasaporte Corriente: pasaporte otorgado a las personas naturales que residan en territorio cubano y necesiten viajar al extranjero por asuntos particulares. También se expide este tipo de pasaporte a las personas que, por razones de servicio a favor de alguna institución, órgano u organismo estatal requieran viajar al exterior. Las solicitudes de estos pasaportes para las personas que decidan viajar por asuntos particulares se tramita mediante las oficinas del Ministerio del Interior, en caso de que sea a solicitud de alguna entidad estatal se tramita en la Dirección de Inmigración y Extranjería. Es válido por dos años y prorrogable por igual cantidad de tiempo hasta un total de 6 años.

Pasaporte de Marino: este pasaporte se expide a todas las personas que sean tripulantes de embarcaciones marítimas que realicen viajes internacionales y se otorga a solicitud del Ministerio de Transporte y del Ministerio de la Industria Alimenticia. La Dirección de Inmigración y Extranjería se encarga de expedir estos pasaportes. Pasaporte válido por un periodo de dos años, con posibilidad de prórroga de dos años más, dos veces consecutivas.

Los pasaportes, según recomendaciones de la Organización de Aviación Civil Internacional (OACI), deben contar con medidas de seguridad que garanticen la identidad de su portador, con el objetivo de evitar la suplantación de identidad o posibles falsificaciones. Las zonas de lectura mecánica es un mecanismo de seguridad recomendado por la OACI, de amplio uso e implementación en pasaportes y documentos de viaje.

1.1.2 Documento de viaje de lectura mecánica.

El uso de zonas de lectura mecánica en los pasaportes como mecanismo de seguridad, proviene desde la década de los años 60. El trabajo en este ámbito es liderado por la OACI que desde 1968 se encarga de elaborar recomendaciones y especificaciones para los pasaportes con zonas de lectura mecánica (ZLM). En 1980 se publica la primera edición del documento 9303 donde se abordan especificaciones y recomendaciones para el pasaporte con zona de lectura mecánica, y que Australia, Canadá y Estados Unidos fueron los primeros en usar como referencia.

El documento de viaje de lectura mecánica (DVLM) se define como: “Documento oficial, conforme a las especificaciones del Doc 9303, expedido por un Estado u organización que el titular emplea en viajes

internacionales (p. ej., pasaporte, visado, documento oficial de identidad) y que contiene datos visuales (lectura ocular) obligatorios y un resumen de datos obligatorio por separado en formato capaz de leerse mecánicamente" (1).

El pasaporte de lectura mecánica (PLM) se define como: "Pasaporte que cumple las especificaciones que figuran en el Doc 9303, Parte 1, Volumen 1 y, opcionalmente, Volumen 2. El PLM está normalmente elaborado en forma de libreta (tamaño ID-3) con páginas que contienen información sobre el titular y el Estado u organización expedidores y páginas para visados y otras anotaciones. La información de lectura mecánica está comprendida en dos líneas de texto OCR-B[2], de 44 caracteres cada una. Estas especificaciones permiten que el PLM tenga forma de tarjeta independiente de tamaño ID-1" (1).

La zona de lectura mecánica (ZLM) se define como un: "Área de dimensiones fijas situada en la página de datos del DVLM que contiene los datos obligatorios y opcionales ordenados de forma que puedan ser leídos mecánicamente con métodos OCR" (1).

La inserción de las zonas de lectura mecánica en los documentos de viaje ha propiciado un incremento de la seguridad en este ámbito. La estandarización de las normas y recomendaciones, por parte de la OACI, ha influido en un mejor control del flujo de personas en las distintas terminales de viajes internacionales a nivel mundial además de proveer soluciones eficaces a los problemas en la confección de documentos.

Los trámites de solicitudes de pasaportes, proveniente de las entidades estatales, contiene información personal del ciudadano; esta información es procesada por el SUIN para comprobar la identidad del ciudadano y permitir la confección del documento de viaje.

1.1.3 Tramitación de pasaportes.

Los procesos de trámite abarcan varios estados que en conjunto permiten darle solución a algún problema. Según Decreto-Ley 302 del 11 de octubre del 2012, en Cuba la tramitación de pasaportes se realiza mediante el Ministerio de Relaciones Exteriores y el Ministerio del Interior, a favor de cualquier ciudadano cubano que solicite su pasaporte o de la institución, órgano u organismo estatal que lo requiera.
Trámite se define como: "Paso de una parte a otra, o de una cosa a otra, cada uno de los estados y diligencias que hay que recorrer en un negocio hasta su conclusión" (6). En el ámbito de la gestión y emisión de pasaportes el trámite necesario para culminar con éxito la entrega del pasaporte consta de varias etapas con disímiles características en las cuales se capturan y procesan los datos de las personas para ser registrados.

[2] Optical Character Recognition (Reconocimiento de Carácter Óptico con fuente B).

El proceso de trámites consulares contempla: los pasaportes corrientes, procedentes de los consulados cubanos en el exterior y los pasaportes diplomáticos y de servicios procedentes del MINREX. Para solicitudes de pasaportes corrientes procedentes de los consulados cubanos, la información actual se capta y registra en el EMIPAS[3] para su remisión al SUIN; se registran los datos de la solicitud, tipos de solicitud, datos identificativos del solicitante y datos de ubicación biométrica. De las solicitudes de pasaporte diplomático y de servicio procedentes del MINREX se registran los datos de las solicitudes, tipos de pasaportes, tipos de solicitud y datos de ubicación biométrica. El procedimiento que se aplica para la tramitación de la solicitud por la oficina a cargo de la captura de datos es: el registro automático en el sistema consular de los datos y posteriormente el envío desde el sistema consular o sistema MINREX de forma digital al SUIN; al recibirse esta solicitud para el proceso de personalización el SUIN verifica la información (7).

1.1.4 Criptografía.

La seguridad en la gestión de la información, principalmente en los trámites de pasaportes, es un tema que siempre está latente en este ámbito. Existen diversas formas para garantizar la seguridad en la gestión de los datos, el uso de algoritmos de encriptación o cifrado, es una de las vías más usadas para el traslado y manejo de información sensible. “La encriptación es el proceso para volver ilegible información considera importante. La información una vez encriptado sólo puede leerse aplicándole una clave” (8).

Los algoritmos de encriptación se encuentran divididos en simétricos y asimétricos. Ambos sistemas ejecutan complejas fórmulas matemáticas para descifrar y se usan claves o llaves, como parámetro de seguridad. Los sistemas criptográficos simétricos o de clave privada son los que utilizan la misma clave para cifrar y descifrar. Estos tienen como inconveniente que la clave debe estar presente en el emisor y en el receptor, creando la interrogante de cómo transferir dicha clave de forma segura (9).Los sistemas asimétricos o de clave pública manejan una doble clave o llave conocidas como clave privada y clave pública. Con la pública se cifra la información y solo con la privada se descifra esa información. Estas llaves permiten su intercambio de forma tal que una sirve para cifrar la otra, al tiempo que es posible que el conocimiento de la clave pública no permita descubrir la privada. Mediante los sistemas asimétricos la información puede ser transferida por medios no seguros puesto que solo la clave pública es la que viaja y que pudiera ser de dominio público pero solo es efectiva si se conoce la clave privada, la cual no viaja.

[3] Sistema Cubano de Emisión de Pasaportes.

1.1.4.1 Algoritmos simétricos.

A continuación se describen algoritmos simétricos (10).

DES[4]

Es un algoritmo de cifrado originalmente creado por IBM a solicitud del *National Bureau of Standard* (Oficina Nacional de Estandarización) y adoptado por el gobierno de los Estados Unidos en 1977 como estándar de cifrado de todas las informaciones sensibles. Este algoritmo tiene como principio de funcionamiento el trabajo en bloques de 64 *bits,* de los cuales 8 *bits* (que representan 1 *byte*) se utilizan para el control de paridad (verificación de integridad de la clave), cada uno de los *bits* de la clave de paridad es utilizado para controlar los *bytes* de la clave de paridad impar. La clave posee una longitud útil de 56 *bits*, el algoritmo realiza combinaciones, sustituciones y permutaciones con el texto y la clave teniendo en cuenta que las operaciones sean reversibles. La clave se codifica en 64 *bits* y está compuesta en 16 bloques de 4 *bits.*

TripleDES

El algoritmo TripleDES posibilita el encadenamiento de tres cifrados a través del uso de dos claves. En 1990 se desarrolla el criptoanálisis diferencial donde se buscaban pares de textos cifrados y planos, con funcionamiento de 15 rondas y con claves de 256 *bits* que ofrecen un gran número de posibilidades pero con el inconveniente de que muchos procesadores ejecutándose al mismo tiempo pueden encontrar la clave correcta. En este sentido TripleDES ofrece el aumento significativo de la seguridad del DES pero consumiendo mayores recursos, se basa en tres iteraciones sucesivas del algoritmo DES consiguiendo una longitud de clave de 128 *bits* debido a que si se cifra el mismo bloque dos veces con dos llaves diferentes se incrementa el tamaño efectivo de la llave.

AES[5]

Este algoritmo fue adoptado en Noviembre de 2001, después de someterse a un proceso de estandarización que duró cinco años, y efectivo después de Mayo de 2002. También es conocido como Rjindael, el cual es sustituto del algoritmo DES, posee un tamaño de bloque fijo de 128 *bits* y compatible con llaves de 128, 192 y 256 *bits,* los cálculos que efectúa son en un campo límite y opera en una matriz de 4x4 *bytes.* El algoritmo posee una alta fiabilidad en cuanto a parámetros de confidencialidad e integridad. Está basado en permutaciones y sustituciones mediante reordenamientos de datos y reemplazo de una unidad con otro, se aplican varias técnicas para ejecutar las permutaciones y sustituciones (11). El algoritmo AES es útil cuando se requiere cifrar y enviar información sensible por canales inseguros, es utilizado por distintos gobiernos y agencias de seguridad a nivel mundial.

[4] *Data Encryptation Standard* (Estándar de Cifrado de Datos).
[5] *Advanced Encryptation Standard* (Algoritmo de Encriptación Avanzado).

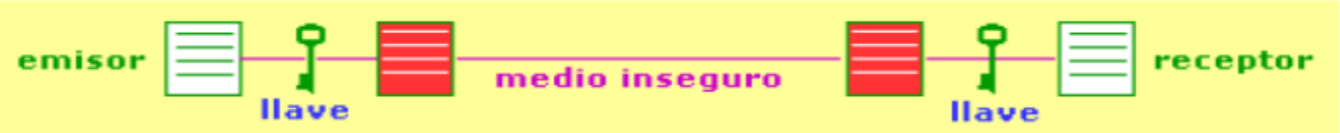

Ilustración 1: Ejemplo de cifrado simétrico *(10)*.

1.1.4.2 Algoritmos asimétricos.

A continuación se describen algoritmos asimétricos (10).

ElGamal

El algoritmo está basado en problemas matemáticos de logaritmos discretos, compuesto por tres partes generador de claves, algoritmo de cifrado y descifrado. Definido por Taher Elgamal en 1984, de amplio uso en software GNU/Linux y sistemas criptográficos, no utiliza ninguna licencia y basado en la intratabilidad computacional del problema del logaritmo discreto. Utilizado en el desarrollo del *Digital Signature Standard,* una característica particular del algoritmo es que utiliza para el cifrado la clave pública del receptor y la clave privada del emisor. Un tercero puede suplantar la firma si encuentra la clave secreta o problemas con la función *hash*, ambos problemas son de difícil ocurrencia. El firmante debe tener cuidado y seleccionar la clave diferente de forma aleatoria para cada firma.

RSA[6]

Algoritmo de encriptación asimétrica creado en 1978 en el Instituto Técnico de Massachusetts (MIT por sus siglas en inglés), el nombre viene dado por las siglas de las iniciales de los apellidos de sus creadores. Es el criptosistema asimétrico más conocido y usado a nivel mundial y el más rápido de todos, sus creadores se apoyaron en el artículo de *Diffie-Hellman* sobre sistemas de llave pública. RSA permite claves de distintas longitudes, no obstante se recomiendan claves con longitudes mayores que 1024 *bits*. La seguridad del algoritmo radica en ser una función computacionalmente segura, debido a que en su funcionamiento realizar la exponenciación modular es fácil pero la operación inversa, la extracción de raíces de modulo Ø, no es práctico sin conocer la factorización de e que es llave privada del sistema. Posee las ventajas de los sistemas asimétricos incluyendo la firma digital.

Ilustración 2: Ejemplo de cifrado asimétrico *(10)*.

[6] *Riverst, Shamir* y *Adleman*.

1.1.4.3 Sistemas mixtos.

Los criptosistemas mixtos son de amplio uso puesto que permite utilizar las principales ventajas de los sistemas simétricos y asimétricos. En estos sistemas la llave pública del receptor es usada para cifrar la llave simétrica que será usada en la comunicación cifrada. Los sistemas asimétricos se utilizan en el envío de la clave y los sistemas simétricos son usados para cifrar los datos.

1.1.4.4 Funciones *Hash*.

Las funciones *hash* son funciones matemáticas que producen un resultado de longitud fija. “Una función resumen (*hash*, en inglés), proporciona una secuencia de bits de pequeña longitud, que va asociada al mensaje, y que debe resultar muy difícil de falsificar” (9). No son funciones reversibles y a diferencia de los sistemas de cifrado que garantizan la confidencialidad, estas funciones solo garantizan la integridad de la información permitiendo asegurar que no se exista suplantación de datos. El resultado obtenido de aplicar la función hash es un conjunto de caracteres alfanuméricos y se produce un resultado de longitud fija y si algún bit se modifica o cambia, el resultado es completamente diferente.

MD5[7]

El algoritmo brinda una reducción criptográfica que garantiza procesar mensajes de entrada de bloques de 512 *bits* y produce salidas de 128 *bits*. De amplio uso y aceptación es utilizado para verificar la integridad de la información creando una síntesis del mensaje que puede ser tan única como la huella dactilar. Algoritmo desarrollado por el profesor Ronald L. Rivest del MIT, diseñado para usarse en la firma digital. Al igual que todas las funciones hash MD5 no es reversible por lo que al convertir un mensaje en una cadena fija de dígitos no hay proceso inverso y se le conoce como resumen del mensaje.

1.1.4.5 Análisis de los métodos criptográficos.

El análisis de los sistemas criptográficos descritos anteriormente y del entorno de ejecución donde será desplegada la propuesta de solución, además del proceso de trámites de pasaporte que se pretende informatizar permitió concluir que la solución más factible, en cuanto a la criptografía para el cifrado de la información que componen las solicitudes de trámites, es la aplicación de un sistema mixto. La selección de usar ambos sistemas, simétricos y asimétricos, se fundamenta por el hecho de que los sistemas asimétricos son más lentos en la ejecución del cifrado y descifrado, y requieren mayor capacidad de procesamiento debido a que son más complejos; mientras que los sistemas simétricos poseen mayor rapidez en el cifrado y descifrado de volúmenes de datos, por lo que se necesita de una infraestructura más simple y la tecnología computacional promedio con que cuentan las entidades estatales no ofrece un óptimo procesamiento para cifrar grandes volúmenes de datos con los sistemas asimétricos. El cifrado del volumen de datos, con mayor

[7] *Message Digest algorithm* (algoritmo de resumen del mensaje 5).

velocidad de ejecución, se efectuará con el sistema simétrico y el cifrado de la llave será mediante la utilización del sistema asimétrico. El algoritmo MD5 será utilizado para comprobar la integridad de las contraseñas de acceso al sistema propuesto.

1.2 Análisis de soluciones nacionales e internacionales.

La tramitación de pasaportes es un proceso común a nivel nacional e internacional, por tal motivo la utilización de soluciones informáticas que lo respalden contribuye significativamente a mejorar la ejecución de dicho proceso. La implantación de estos sistemas posibilita contar con la informatización del proceso, mejorar los sistemas de registro de los ciudadanos y los parámetros de identificación. En el siguiente epígrafe se abordan soluciones nacionales y extranjeras que permiten efectuar trámites de solicitudes de pasaportes, sus características y funcionalidades.

1.2.1 Sistema Único de Identificación Nacional (SUIN).

El SUIN es un sistema informático de factura nacional, elaborado por el Centro de Identificación y Seguridad Digital, perteneciente a la Universidad de las Ciencias Informáticas (UCI), de conjunto con la Dirección de Identificación, Inmigración y Extranjería del Ministerio del Interior. Esta herramienta surge dada la necesidad del Estado Cubano de contar con la informatización de los procesos referentes a la identificación y registro del ciudadano, además de la no existencia de políticas bien definidas para garantizar la confiabilidad en el intercambio de información y de establecer la biometría como un parámetro de identificación segura y confiable.

Con la implantación del SUIN se logran sentar las bases para el uso de la biometría y de un único número de identificación para los ciudadanos cubanos. Ofrece además el rediseño de los procesos facilitando la integración de bases de datos y sistemas acordes con las normas y legislaciones vigentes en el país. El sistema está diseñado para ejecutarse en el sistema operativo *Microsoft Windows* XP SP3[8] o superior y *Windows Server* 2008 R2, en el entorno de desarrollo *Visual Studio Team System* 2010, lenguajes de programación C#, *JavaScript*, ASP.NET[9], HTML, CSS, utiliza el *Microsoft* .NET *framework* 3.5, el servidor de aplicaciones *Internet Information Server* 7.5 y el gestor de base de datos *Oracle* 11g R2 (4).

Mediante el sistema es posible tramitar las solicitudes de pasaportes realizadas por los ciudadanos cubanos que desean viajar al exterior por asuntos particulares. El Pasaporte Corriente es el único de los cinco tipos de pasaporte que es posible confeccionar mediante el uso del SUIN, permitiendo que todas las ventajas que posee el sistema sean aplicables al trámite del pasaporte. El uso de parámetros biométricos, el chequeo

[8] *Service Pack* 3 (Paquete de Servicios, tercero de su tipo).
[9] *Active Server Pages* (Servidor de Páginas Activo).

contra la base de datos única del ciudadano y la gestión informática del proceso de captura y manejo de los datos posibilita la elaboración del Pasaporte Corriente en un menor tiempo y con mayor seguridad en el proceso de emisión.

1.2.2 Servicio Administrativo de Identificación, Migración y Extranjería (SAIME).

El sistema SAIME surge de la colaboración entre Cuba y la República Bolivariana de Venezuela con el objetivo de mejorar el servicio de identificación, registro y migración de los ciudadanos. La misión de este sistema es mejorar la identificación ciudadana haciendo uso de altas tecnologías en el proceso para "…garantizar oportunamente el derecho a la identidad y seguridad jurídica, así como el ejercicio de sus atribuciones de migración…" (12) y efectuar un alto control sobre los procesos de migración y extranjería.

Con este sistema se pretende "Ser un Servicio de referencia y de sólido prestigio a nivel nacional e internacional por la excelencia en la gestión que presta en las áreas de identificación, migración y extranjería…" (12). El SAIME permite al usuario realizar la solicitud de inicio de trámite de pasaportes desde la web de manera no presencial y que la estancia del mismo en las oficinas, no requiera de todas las etapas que abarca el proceso. La aplicación está desarrollada en *Visual Studio* 2003, utilizando el gestor de base de datos Oracle 10gR2 y mediante el empleo de la metodología RUP. Establece el uso de las normas y estándares recomendados por la OACI para la confección de pasaportes.

1.2.3 Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de Venezuela (SEDIP).

El Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela está desarrollado conjuntamente por el Ministerio del Poder Popular para Relaciones Exteriores (MPPRE) de Venezuela y la UCI. El objetivo del sistema es eliminar los procesos deficientes, que eran realizados de forma manual, mejorar los mecanismos de seguridad que no son suficientes y no gestionan de forma eficaz toda la información necesaria sobre los titulares (13).

Existen implementados tres subsistemas con módulos integrados que brindan la opción de realizar solicitudes en línea, captura de datos, supervisión del trámite, personalización, control de calidad y registro de la entrega de los documentos aportando eficiencia en el tratamiento de los datos e imágenes de los titulares.

Los módulos que componen el sistema se encuentran distribuidos en los subsistemas de Solicitud, Captación y supervisión y Personalización. El subsistema de Solicitud está divido en dos aplicaciones web

desarrolladas en PHP[10] y una aplicación de escritorio que brindan diversas funcionalidades asociadas a los tramites de emisión de pasaportes diplomáticos. El subsistema de Captación y supervisión es una aplicación de escritorio desarrollada en JEE[11] que posibilita la gestión de roles, usuarios, configuración de parámetros, captar datos y obtener reportes. El subsistema Personalización es una aplicación de escritorio desarrollada en JEE que brinda la posibilidad de ejecutar la personalización de los documentos de pasaportes, validar la calidad de los documentos, mantener un control de los recursos y obtención de reportes.

1.2.4 Análisis de resultados.

El SUIN permite la gestión del pasaporte Corriente, aportando disímiles ventajas en la confección de este tipo de pasaporte tales como el chequeo de rasgos biométricos y chequeo contra la base de datos única del ciudadano pero solo para este tipo de pasaporte además de estar desarrollado mediante herramientas privativas solo válidas en el sistema operativo *Windows*.

El SAIME permite que los usuarios realicen el pedido de solicitudes de trámites de pasaportes sin estar presentes en las oficinas, brindando ventajas positivas en el proceso; pero este sistema trabaja desde la web de una manera personalizada para la República Bolivariana de Venezuela, con mayores posibilidades de conectividad e infraestructura tecnológica, además de estar desarrollado bajo herramientas privativas y solo válidas en el sistema operativo *Windows*.

El Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de Venezuela brinda la posibilidad de eliminar procesos deficientes en la tramitación de pasaportes, mejorar los mecanismos de seguridad y gestionar de forma eficaz toda la información necesaria sobre los titulares pero este sistema trabaja desde la web, con herramientas privativas, personalizado para la República Bolivariana de Venezuela con mayores posibilidades de conectividad e infraestructura tecnológica y adaptado al entorno diplomático de dicho país.

Las soluciones estudiadas ofrecen ventajas significativas al proceso de trámites de pasaporte pero la problemática actual plantea la necesidad de contar con un sistema multiplataforma, desarrollado mediante herramientas con filosofía de software libre y que no trabaje desde la web. Debido a que el país está inmerso en la migración hacia software libre y dos de las soluciones analizadas trabajan en software privado se determina que el uso de estos sistemas no es factible, además de analizar que la conectividad y la infraestructura tecnológica disponibles no son adecuadas para trabajar con el sistema desde la web. Por

[10] Acrónimo recursivo que significa *PHP Hypertext Pre-processor* (Pre-procesador de hipertexto).
[11] *Java Enterprise Edition* (Java Edición Empresarial).

tanto se determina que las soluciones nacionales e internacionales analizadas no son aplicables a la problemática.

1.3 Herramientas y metodologías a utilizar en la propuesta de solución.

En el desarrollo del sistema de gestión de datos, planteado inicialmente como objetivo, se hace necesario el estudio y posterior uso de herramientas y tecnologías acordes al entorno de desarrollo. Las aplicaciones seleccionadas poseen características y funcionalidades que garantizan su utilización en la implementación de la propuesta de solución y brindan apoyo al proceso de soberanía tecnológica en los que se encuentra inmerso la Universidad de las Ciencias Informáticas (UCI) y Cuba, a continuación se abordan estas principales funciones y características de los patrones arquitectónicos, herramientas, metodologías y entornos de desarrollo seleccionados.

1.3.1 Metodología de desarrollo de software.

"Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo" (14). Las metodologías de desarrollo de software permiten la gestión y administración de un proyecto con el objetivo terminar el ciclo de vida con las más altas posibilidades de éxito. Detallan los artefactos, roles, actividades involucradas, prácticas y técnicas recomendadas (15).

Según la filosofía de desarrollo las metodologías de desarrollo se pueden clasificar en dos grupos (15):

- **Metodologías tradicionales (no ágiles):** regidas por una fuerte planificación durante el proceso de desarrollo, llevando a cabo una intensa etapa de análisis y diseño antes de construir el sistema. Centradas en tener un amplia documentación del proyecto y en cumplir con el plan propuesto.
- **Metodologías ágiles:** el proceso es ágil cuando el desarrollo es incremental, permitiendo entregas pequeñas del producto con ciclos rápidos; cooperativo, debido a que existe estrecha colaboración entre el cliente y los desarrolladores; sencillo porque es fácil de aprender y modificar; y adaptable debido a que es ajustable a los cambios repentinos.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo	Impuestas externamente

de desarrollo)	
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1: Tabla comparativa entre metodologías ágiles y tradicionales *(16)*.

- **Metodología tradicional.**

1.3.1.1 *Rational Unified Process.*

Las metodologías de desarrollo tradicional se rigen por una extensa documentación y planificación de las actividades que están definidas. Durante el proceso se realiza un fuerte análisis y diseño de la implementación que se tiene planeada. Estas metodologías establecen un control riguroso sobre las actividades del proyecto con una especificación precisa en cuanto a los requerimientos y el modelado. La disciplina, el énfasis que se estable los aspectos de cada fase, la definición de roles y la poca adaptación que efectúan ante los cambios son algunas de las características más destacadas. De las metodologías tradicionales la más conocida es *Rational Unified Process* (RUP por sus siglas en inglés).

RUP posee diversas características apropiadas para entornos de desarrollo, con grandes equipos y donde los cambios en el desarrollo no son comunes. Está basado en componentes e interfaces definidas, es posible especializarlo para diversos sistemas y ofrece un óptimo desempeño con el Lenguaje Unificado de Modelado (UML[12] por sus siglas en inglés).

[12] Unified Modeling Language (Lenguaje de Modelado Unificado).

- **Metodologías ágiles.**

1.3.1.2 *eXtreme Programming*.

La aplicación de este desarrollo ágil viene dado por las principales características que lo acompañan y las opciones que ofrece como proceso. En 2001 Kent Beck y otros desarrolladores firmaron lo que se conoce como el "Manifiesto para el desarrollo ágil de software" lo cual plasmaba:

"Hemos descubierto mejores formas de desarrollar software al construirlo por nuestra cuenta y ayudar a otros a hacerlo. Por medio de este trabajo hemos llegado a valorar: A los individuos y sus interacciones sobre los procesos y herramientas. Al software en funcionamiento sobre la documentación extensa. A la colaboración del cliente sobre la negociación del contrato. A la respuesta al cambio sobre el seguimiento de un plan" (17).

Características de la *eXtreme Programming* (18)**:**

- Colaboración directa con el cliente.
- Habilidad de respuesta al cambio y adaptabilidad.
- Apropiada para equipos de desarrollo pequeños.
- Permite retroalimentación continua.
- Planificación a corto plazo

Actividades de XP (17)**.**

- **Planeación:** esta actividad tiene su inicio con la creación de las historias de usuarios en donde son descritas las características y requerimientos de la aplicación. Estas historias de usuarios son elaboradas por los clientes y él mismo le asigna una prioridad. En esta etapa los clientes y los desarrolladores actúan en conjunto para decidir cómo ordenar las historias de usuarios con vista al desarrollo, la fecha de entrada y otros aspectos de interés.
- **Diseño:** en esta actividad se trata de seguir la idea de mantener lo más simple posible, debido a que es preferible un diseño simple y no uno complejo. En etapa de XP el uso de tarjetas CRC (colaboración, responsabilidad, clase) es apoyado debido a la ayuda que ofrecen en la identificación y organización de las clases. En XP el diseño puede suceder tanto antes como después de la codificación.
- **Codificación:** antes de entrar en esta actividad es recomendado realizar pruebas de unidad donde se entrenen las historias de usuario. Es preferible que en el desarrollo de esta fase se aplique la característica de programación por parejas donde se trabaje con cada historia de usuario permitiendo el uso de más conocimiento y asegurando calidad. En esta actividad los desarrolladores deben estar enfocados en el problema actual.
- **Prueba:** en esta actividad se denota como las pruebas de unidad son importantes en la ejecución previa a la codificación, por lo que estas deben implementarse en un entorno de trabajo para apoyar la estrategia de regresión de prueba. Las pruebas de integración y validación se pueden ejecutar de

forma diaria propiciando un método de alerta temprana. En XP las pruebas de aceptación son especificadas por el cliente, están orientadas a las características y funcionalidades del sistema y son derivadas de las historias de usuario.

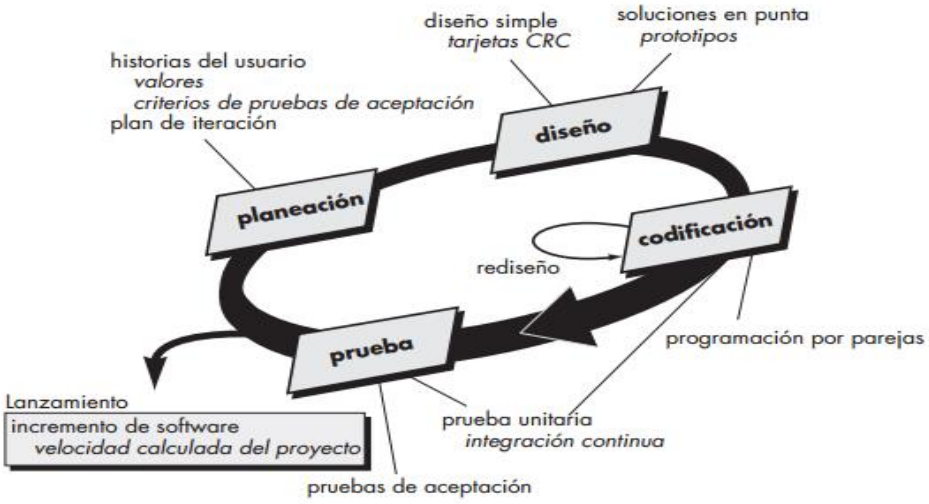

Ilustración 3: Actividades de la eXtreme Programming *(18)*.

1.3.1.3 SCRUM.

La metodología de desarrollo SCRUM[13], creada por Jeff Sutherland y su equipo de desarrollo al comienzo de los años 90, guía sus principios según el manifiesto ágil y es utilizado para regir actividades de desarrollo dentro de procesos de análisis que incorporan las actividades estructurales: requerimientos, análisis, diseño y evolución. En las actividades definidas las tareas suceden con un patrón denominado *sprint*, la cantidad de estos patrones varían según la complejidad y tamaño del producto**.** SCRUM resalta la utilización de patrones de proceso de software para proyectos con cortos plazos de entrega, requerimientos cambiantes y negocios críticos (18).

Patrones del proceso (18)**.**

- **Retraso:** lista de prioridades de los requerimientos o características del proyecto.
- ***Sprint*:** unidades de trabajo necesarias para alcanzar un requerimiento definido en el retraso.
- **Reuniones SCRUM:** reuniones cortas, de aproximadamente 15 minutos, que se efectúan a diario.
- **Demostraciones preliminares:** entregas del incremento del software al cliente para que las funcionalidades implementadas sean mostradas y sean evaluadas por él mismo.

[13] Nombre que procede de una jugada que se aplica en partidos de *Rugby*.

1.3.1.4 Cristal (*Crystal Methodologies*).

La metodología cristal(o metodologías cristal), es creada por Alistar Cockburn con el objetivo de tener un enfoque de desarrollo que ofrezca maniobrabilidad durante lo que se caracteriza como “un juego cooperativo con recursos limitados, de invención y comunicación, con el objetivo primario de entregar software útil que funcione y con la meta secundaria de plantear el siguiente juego”. Para obtener la maniobrabilidad se definieron un conjunto de metodologías, con elementos comunes a todos pero con roles, patrones de procesos, productos y prácticas que son únicas en cada uno (18).

Cristal está centrada en las personas, ya que de ellos depende el éxito, y en la reducción al máximo del número de artefactos. El equipo de desarrollo es clave por lo que es necesario invertir esfuerzos en la mejora de las habilidades y tener políticas bien establecidas, dependiendo del tamaño del equipo (*Crystal Clear* establece de 3 a 8 miembros y *Crystal Orange* de 25 a 50 miembros) (16). Las metodologías Cristal no especifican un ciclo de vida fijo o modelo de procesos sino que determina una guía de políticas estándar, productos, asuntos locales, herramientas y roles (19).

Prácticas de las metodologías Cristal (19)**.**

- Planificación por etapas.
- Revisiones y resúmenes.
- Monitorización.
- Paralelismo y flujo.
- Estrategia de diversidad integral.
- Técnicas de puesta a punto de la metodología.

1.3.1.5 Análisis de las metodologías.

Para seleccionar la metodología para el desarrollo del software, se realizó un análisis de las características y funcionalidades de las metodologías descritas en los epígrafes anteriores, teniendo en cuenta además cómo se realiza el proceso de trámites de pasaportes, el contexto actual del problema y características del equipo de desarrollo. Debido a que el equipo de desarrollo es pequeño (dos desarrolladores), a que se necesita colaboración directa con el cliente (jefe de proyecto Identidad Cuba), a que se necesita la menor documentación posible y solo los artefactos necesarios, además de que el proyecto no presenta gran complejidad, se concluye que utilizar una metodología tradicional no es la opción más adecuada. Se determinó que la utilización de las metodologías tradicionales no resulta factible debido a que plantean determinada resistencia al cambio, procesos más controlados, la presencia del cliente solo en reuniones, grandes grupos de desarrollo, más artefactos y más roles. En este sentido se selecciona una metodología ágil como guía para el proceso de desarrollo del software.

Respecto a las metodologías ágiles se concluye que:

- XP brinda facilidad respecto a la integración del cliente al proceso, mantiene la idea de tener simple la presentación, favorece la retroalimentación continua y la existencia de pocos roles. Ofrece una calidad del producto a través de la satisfacción del cliente y de la conformidad a los requerimientos; posee resultados positivos en el tratamiento de la calidad además de demostrar prácticas y métodos más eficientes en este aspecto. La metodología es aplicable al uso de distintas herramientas para la gestión de proyectos, repositorios y f*ramework* para ejecutar pruebas unitarias.
- SCRUM se enfoca más en la gestión de proyectos y no presenta prácticas concretas para asegurar la calidad debido a que su propósito es sacar adelante el proyecto. Esta metodología está diseñada para realizar iteraciones de 30 días, denominadas *sprints*, y efectuar reuniones diarias de 15 minutos lo cual requiere de práctica para lograr adaptación y lograr el máximo de provecho. SCRUM en su modelo de proceso no específica las fases de diseño, codificación y pruebas unitarias debido a que lo considera al final del *sprint*.
- Cristal es una metodología que se considera abstracta debido a que no se establece como desarrollar artefactos. Orienta sus parámetros de calidad en procesos de comunicación por lo que no asegura calidad. En el ciclo de vida del proyecto abarca pocas fases y demanda la creación de artefactos en otras fases. Cristal aplica una guía prescriptiva por lo que las prácticas, procesos y métodos deben seguirse como se definen; mientras que otras metodologías ágiles son más ajustables.

	Crystal	**Scrum**	**XP**
• **Sistema como algo cambiante**	4	5	5
• **Colaboración**	5	5	5
• **Resultados**	5	5	5
• **Simplicidad**	4	5	5
• **Adaptabilidad**	5	4	3
• **Excelencia técnica**	3	3	4
• **Prácticas de colaboración**	5	4	5
Media características metodologías	4.4	4.2	4.4
Media Total	4.5	4.7	4.8

Tabla 2: Ranking de agilidad entre metodologías ágiles (los valores más altos representan mayor agilidad) *(16)*.

Por lo anteriormente expuesto se determina que la *eXtreme Programming* es la metodología ágil que se adapta mejor a las necesidades del proceso de desarrollo, teniendo como premisa el trabajo en equipos pequeños, colaboración directa con el cliente, poca documentación y ajustable a los cambios.

1.3.2 Sistema Gestor de Base de Datos.

1.3.2.1 PostgreSQL.

Es un potente, y ampliamente usado, gestor de bases de datos objeto-relacional que aplica el modelo cliente-servidor en su sistema. Esta herramienta posee entre sus características el uso de multiprocesos y no multihilos certificando así mayor estabilidad. PostgreSQL es, en la actualidad, la base de datos basada en código libre más potente y avanzada a nivel internacional (20).

Características en la programación.

- Funciones almacenadas en numerosos lenguajes de programación (PL/pgSQL, PL/Perl, PL/Python y PL/Tcl).
- Numerosos tipos de datos y posibilidad de definir nuevos tipos.
- Soporte para almacenamiento de objetos binarios grandes.

Características Generales de PostgreSQL (20)**.**Es una base de datos 100% ACID[14].

- Integridad referencial.
- Copias de seguridad en caliente.
- *Unicode.*
- Múltiples métodos de autenticación.
- Acceso encriptado vía SSL[15].
- Completa documentación.
- Licencia BSD[16].

Características SQL[17] del gestor de bases de datos.

- Llaves primarias y llaves foráneas.
- SQL92, SQL99, SQL2003, SQL2008.
- Columnas auto-incrementales.
- *Joins*[18].
- Vistas.
- Disparadores.

[14] ACID es un acrónimo en inglés para los términos *Atomicity, Consistency, Isolation, Durability* y hace referencia a la evaluación que se realiza sobre el trabajo de las bases de datos y su arquitectura. (35)
[15] *Secure Socket Layer* (Capa de Conexión Segura).
[16] *Berkeley* Software *Distribution* (Distribución de Software de *Berkeley*).
[17] *Structured Query Language* (Lenguaje Estructurado de Consulta).
[18] Término que hace referencia a la unión entre tablas de una base de datos.

1.3.2.2 MySQL.

MySQL es un gestor de base de datos, de código abierto, desarrollado por *Sun Microsystems* y actualmente propiedad de *Oracle Corporation*. Este sistema está distribuido bajo licencia GNU GPL[19] pero si alguna empresa desea incluirlo en sus productos privativos debe adquirir una licencia específica para su uso. *MySQL* soporta varios lenguajes de programación, muchos de ellos de amplio uso, entre ellos C, C++, C#, *Java*, *Python*, PHP, todos estos lenguaje con una interfaz específica para la programación. Este gestor de base de datos es aplicado en entornos de desarrollo web, en diversas plataformas como GNU/Linux, Windows, BSD, Mac OS. Es un sistema diseñado para que la velocidad de lectura sea un parámetro distintivo, ligado estrechamente al lenguaje PHP e ideal para entornos donde exista una amplia lectura de datos (21).

Características de MySQL (21)**.**

- Alta velocidad de conexión.
- Consumo de pocos recursos de *hardware*.
- No hay límites en el tamaño de los registros.
- Soporte completo para operadores y funciones.
- Posibilidad de selección de múltiples motores de almacenamiento.
- Agrupación de transacciones.

1.3.2.3 Análisis de los sistemas gestores de base de datos.

El análisis de los sistemas de base de datos, anteriormente descritos, permite concluir que presentan características técnicas diferentes y se encuentran orientados a diversos aspectos. El gestor de base de datos MySQL posee una alta velocidad de lectura de datos, consumo de pocos recursos de *hardware*, alta integración con el lenguaje de programación PHP y con otros lenguajes de uso extensivo y aplicable en múltiples plataformas; pero está mejor diseñado para ser aplicado en entornos de desarrollo web, y características como integridad y seguridad de los datos representan un punto débil además de ser un sistema en el cual una gran parte del código es propiedad de Oracle. El gestor de bases de datos PostgreSQL posee propiedades y funcionalidades que garantizan su uso dentro del entorno de desarrollo de la propuesta de solución. Es un sistema enfocado en la fiabilidad e integridad de los datos, posee características orientadas al programador, ejecuta un potente planificador de consultas y está diseñado para ser usado en sistemas donde la consistencia de la base de datos es fundamental. El alto porcentaje de evaluación obtenido en las pruebas ACID, la estabilidad de arquitectura, estar creado bajo la filosofía de software libre y convertirse en el gestor más potente de esta tecnología, además de la amplia comunidad

[19] *General Public License* (Licencia Pública General de GNU).

de usuarios en activa colaboración, son características del gestor que posibilitan su inserción en el modelo de desarrollo.

1.3.3 Lenguaje de programación.

El lenguaje de programación que se selecciona es ***Java***, debido a que es ampliamente usado en la elaboración de cientos de aplicaciones a nivel mundial, cuenta con soporte regular y millones de desarrolladores en todo el mundo. Es un lenguaje con múltiples propósitos, orientado a objetos y basado en clases. Su sintaxis deriva de C/C++ y es posible ejecutarlo en cualquier máquina virtual de *Java* (JVM por su siglas en ingles). Los programas desarrollados con este lenguaje necesitan del *Java Development Kit* (JDK) debido a que este software provee las herramientas de desarrollo a ejecutar en la plataforma. **Ventajas del uso de JDK** (22)**.**

- Soporte para lenguaje tipados.
- Chequeo de los archivos de las clases.
- Actualiza la arquitectura del cargador de clases.
- Actualización de clases.

Características de *Java* (23)**.**

- Independencia de la plataforma.
- Basado en estándares.
- Entornos de ejecución coherentes.

Ventajas de *Java* (23)**.**

- Alto rendimiento.
- Fácil de aprender.
- Aplicaciones portátiles con alto rendimiento.

El análisis de las características y ventajas que posee el lenguaje de programación permite concluir que el uso del mismo dentro del desarrollo de la propuesta de solución es factible. *Java* puede ejecutarse con todas las plataformas de trabajo de mayor uso a nivel mundial tanto libres como privadas. Posee un alto rendimiento contando con optimización integrada para varios procesos, es un lenguaje preferido en muchas instituciones educacionales de relevancia en el ámbito internacional, además de ser un entorno de desarrollo de gran popularidad potenciando una amplia comunidad de desarrolladores. El lenguaje brinda modelos de seguridad ideales para su aplicación en entornos de trabajo con bases de datos y en red.

1.3.4 Mapeo Objeto-Relacional.

El trabajo con la base de datos relacional y todo lo que eso implica puede no ser fácil debido al volumen de datos que es posible procesar y al tiempo disponible para el desarrollo. En este sentido ***Hibernate*** es una herramienta que facilita el mapeo objeto/relacional a fin con el desarrollo de la aplicación. El término de

mapeo objeto/relacional (ORM[20]) se refiere: “a la técnica de mapear una representación de datos desde un modelo de objeto a un modelo de datos relacionales con un esquema basado en SQL” (24).

Esta herramienta posibilita el trabajo con las tablas de las bases de datos, consultas y en la recuperación de datos. *Hibernate* está diseñado para abreviar el trabajo de los desarrolladores y ser útil cuando los modelos están orientados a objetos. La utilización del mapeo entre los objetos y la entidad-relación garantiza un mejor análisis y el uso de características propias orientadas a objetos.

Características (24)**.**

- Mapeo objeto/relacional.
- Persistencia idiomática.
- Alto rendimiento.
- Escalabilidad.
- Seguridad.

La aplicación de esta herramienta en el trabajo con el acceso a datos posibilita economizar tiempo en función de lograr el máximo aprovechamiento del tiempo de desarrollo, además que este ORM está desarrollado para trabajo con la plataforma *Java*. La facilidad que brinda en cuanto a la manipulación de información en las tablas, en el desarrollo de clases persistentes, la capacidad de rendimiento, su facilidad de adaptación al entorno y la estabilidad que posee, son propiedades que permiten la utilización de esta herramienta para el mapeo objeto/relacional en la implementación de la propuesta de solución.

1.3.5 Entorno de desarrollo.

1.3.5.1 NetBeans versión 7.4.

Este IDE[21] es una plataforma de desarrollo de aplicaciones *Java* de código abierto, con una amplia comunidad de usuarios en constante crecimiento fundada por *Sun MicroSystems* (actualmente es propiedad de la compañía ORACLE). Desde un principio fue lanzado con soporte *Swing*[22] según las mejoras del JDK 1.3, haciéndolo una alternativa realmente viable para el desarrollo de aplicaciones con *Java*.

Características de NetBeans (25)**.**

- Compatibilidad con HTML5[23].
- Compatibilidad con *JavaScript*.

[20] *Object-Relational Mapping* (Mapeo Objeto Relacional).
[21] *Integrated Development Environment* (Entorno de Desarrollo Integrado).
[22] Biblioteca gráfica para el lenguaje *Java*.
[23] *HyperText Markup Language* (Lenguaje de Marcado de HyperTexto).

- Compatibilidad con CSS[24].
- Soporte para los lenguaje *Java*, C/C++, XML y HTML, PHP, Groovy, Javadoc.
- Aplicación de pruebas unitarias.
- Permite el control de versiones.
- Permite colocar puntos de interrupción en el código fuente.

Con este entorno es posible realizar todas las tareas asociadas a la programación como editar código, compilarlo, ejecutarlo y depurarlo. Brinda un desarrollo rápido y fácil de aplicaciones *Java* para escritorio, móviles y web. Proporciona un excelente apoyo integral para las tecnologías *Java* más recientes y las últimas mejoras de las especificaciones del lenguaje antes de que otros IDE´s, además de diferentes vistas de los datos a partir de múltiples ventanas y herramientas útiles para la creación de aplicaciones. Es el primer IDE libre que ofrece soporte para JDK 8 avanzado, JDK 7, Java EE 7. Proporciona herramientas de análisis estático, especialmente la integración con la herramienta *FindBugs*[25] ampliamente utilizado para identificar y solucionar problemas comunes en el código de Java.

1.3.5.2 *Eclipse.*

Eclipse es un entorno de desarrollo, de código abierto, compuesto por herramientas destinadas al desarrollo de proyectos. Este IDE fue desarrollado por IBM a partir de que en 2001 se creara un consorcio para el desarrollo futuro de Eclipse y en 2003 es creada la fundación Eclipse como entidad independiente. Este programa basa su arquitectura en el desarrollo de lo que se denomina “Arquitectura de Clientes Enriquecidos” o *Rich Client Platfom* constituido por la plataforma principal, la OSGi[26], el Standard *Widget Toolkit* (SWT por sus siglas en inglés) que es un conjunto de componentes o librerías para construir interfaces gráficas, JFace (para manejo de archivos, texto y editores) y el Workbench con vistas, editores, perspectivas y asistentes. Emplea módulos para las distintas funcionalidades definidas en su arquitectura, compatibilidad con lenguajes como C, C++, Python, LaTex, aplicaciones Telnet y gestores de base de datos.

Características de *Eclipse.*

- Compilación en tiempo real.
- Aplicación de pruebas unitarias.
- Permite el control de versiones.
- Integración con Ant[27].
- Asistentes para creación de proyectos.

[24] *Cascading Style Sheets* (Hojas de Estilo en Cascada).
[25] Encontrar errores: Programa de tipo código abierto que busca errores en programas de *Java*.
[26] *Open Service Gateway Initiative* (Iniciativa Abierta de Servicios de Salida).
[27] Herramienta para realizar tareas mecánicas y repetitivas.

1.3.5.3 Análisis de los entornos de desarrollo.

El análisis de estos entornos de desarrollo permite concluir que Eclipse es de gran ayuda para la ejecución de cualquier tipo de aplicación, posee un trabajo basado en la preconfiguración de ventanas y editores, proporciona asistentes y ayudas en la gestión de proyectos, ejecutable en varios sistemas operativos, compatible con diversos lenguaje de programación incluido *Java*, producto de código abierto bajo la licencia pública de Eclipse, incluye un potente depurador de código y extensa compilación de módulos.

Netbeans es aplicable a sistemas operativos con filosofía de software libre y propietario, está diseñado principalmente para el lenguaje de programación seleccionado, posee una amplia comunidad de usuarios y desarrolladores de todo el mundo, es un producto libre y gratuito sin restricciones bajo las licencias GPL[28] y CDDL[29], posee buen editor de código, herramienta de desarrollo modular para una amplia gama de tecnologías de desarrollo de aplicaciones, multilenguaje y facilita la gestión de proyectos además de ser extensible a través de módulos.

Ambos entornos de desarrollo poseen características similares, potentes herramientas internas, trabajan con el lenguaje de programación seleccionado, la gestión de proyectos e interfaces gráficas además de ser multiplataforma, por tal motivo se determina que la selección del IDE viene dado por la familiarización que tenga el equipo de desarrollo con uno u otro programa de acuerdo a las características de la propuesta de solución, quedando seleccionado el *Netbeans* como programa para la implementación de la propuesta; debido a que los programadores poseen mayor experiencia en el trabajo con este entorno y mejor conocimiento de las funcionalidades y características internas permitiendo además que no existan retrasos en el proceso de desarrollo por cuestiones de adaptación y entendimiento al trabajo con *Eclipse*.

1.3.6 Herramienta de modelado.

La ejecución de modelado del sistema se realiza en la herramienta CASE[30] **Visual Paradigm 8.0**. *Visual Paradigm for* UML (VP-UML) es una herramienta de diseño de software concebida para proyectos que necesiten modelar lenguajes y diagramas (26).Esta herramienta garantiza el desarrollado de aplicaciones utilizando el Lenguaje de Modelado Unificado. El uso de la aplicación es idóneo para quienes necesiten diseñar y desarrollar sistemas confiables y estables orientados a objetos.

Características de Visual Paradigm (26)**.**

- Navegación intuitiva.
- Potente generador de informes en formato PDF[31]/HTML.
- Entorno superior de modelado.

[28] *General Public License* (Licencia Pública General de GNU).
[29] *Common Development and Distribution License* (Licencia Común de Desarrollo y Distribución).
[30] *Computer Aided* Software *Engineering* (Ingeniería de Software Asistida por Computadora).
[31] *Portable Document Format* (Formato de Documento Portable).

- Sincronizador de código fuente.

1.3.6.1 Lenguaje de modelado.

El lenguaje de modelado seleccionado es **UML 2.1**, el cual es la especificación más usada a nivel mundial para describir estructuras de aplicaciones, comportamientos, procesos de negocio y estructuras de datos. Este lenguaje ofrece visualización de esquemas de software utilizando cualquiera de las herramientas disponibles al tiempo que es posible utilizar alrededor de trece tipos de diagramas divididos en tres categorías donde se representan la estructura de la aplicación, el comportamiento y las interacciones. Es posible modelar casi todo tipo de sistema flexible, vinculado a lenguajes como *Java* y C# además de componer modelos transaccionales en tiempo real. Permite su utilización como lenguaje de modelado independientemente de la metodología de software utilizada pues posee metodología propia (27).

1.4 Conclusiones del capítulo.

- Los trámites para las solicitudes de pasaportes están asociados al control y chequeo de parámetros biométricos.
- Las soluciones estudiadas, aunque aportan ventajas significativas al proceso de trámite de pasaporte, su aplicación en las entidades pertenecientes al sector estatal no resulta factible debido a que las herramientas de desarrollo son privada, las licencias de uso representan un alto costo además de que no están diseñadas para satisfacer las necesidades de la problemática.
- La selección de la metodología y herramientas permitirán implementar la propuesta de solución acorde a los requerimientos definidos además de ofrecer una guía para el proceso.

Capítulo 2. Características, análisis y diseño.

En el presente capítulo se abordan los temas relacionados con la descripción del proceso de negocio y modelos de dominio para un mejor análisis y entendimiento del proceso de solicitud de trámite de pasaporte. Se describen los requerimientos funcionales y no funcionales del sistema, los cuales se representan según los artefactos que propone la metodología XP.

2.1 Descripción del problema.

En las entidades estatales de Cuba, el proceso de solicitudes de trámites de pasaportes se lleva a cabo mediante la gestión manual de la información. El proceso no cuenta con las ventajas que se tienen implementadas para la confección del Pasaporte Corriente debido a que este último se gestiona mediante un sistema informático, con el cual es posible aplicar chequeo biométrico y verificación a través de la base de datos única del ciudadano. También el proceso ejecutado en las entidades estatales tiene como desventaja que la información hacia las oficinas expedidoras no es realizada de manera segura, pudiendo existir pérdidas y/o modificación de los datos personales capturados. El desarrollo actual del proceso de trámites, es lento debido a cómo se trabajan las solicitudes de pasaportes y esto afecta las necesidades de las organizaciones. Por tal motivo se pretende desarrollar un sistema informático capaz de garantizar la gestión de datos para la tramitación no presencial de pasaportes, en Cuba, que cumplan con los estándares internacionales en materia de documentos de viaje.

2.1.1 Modelo del dominio.

El modelo del dominio brinda la posibilidad de representar, de un modo estático, el entorno donde se desarrolla el flujo de eventos del proceso definido. Mediante un diagrama es posible representar las relaciones que existen entre los objetos que interactúan en el proceso. El objetivo del modelo de dominio es colaborar con el entendimiento de las definiciones utilizadas por los usuarios y con las que se trabajarán durante el proceso de desarrollo.

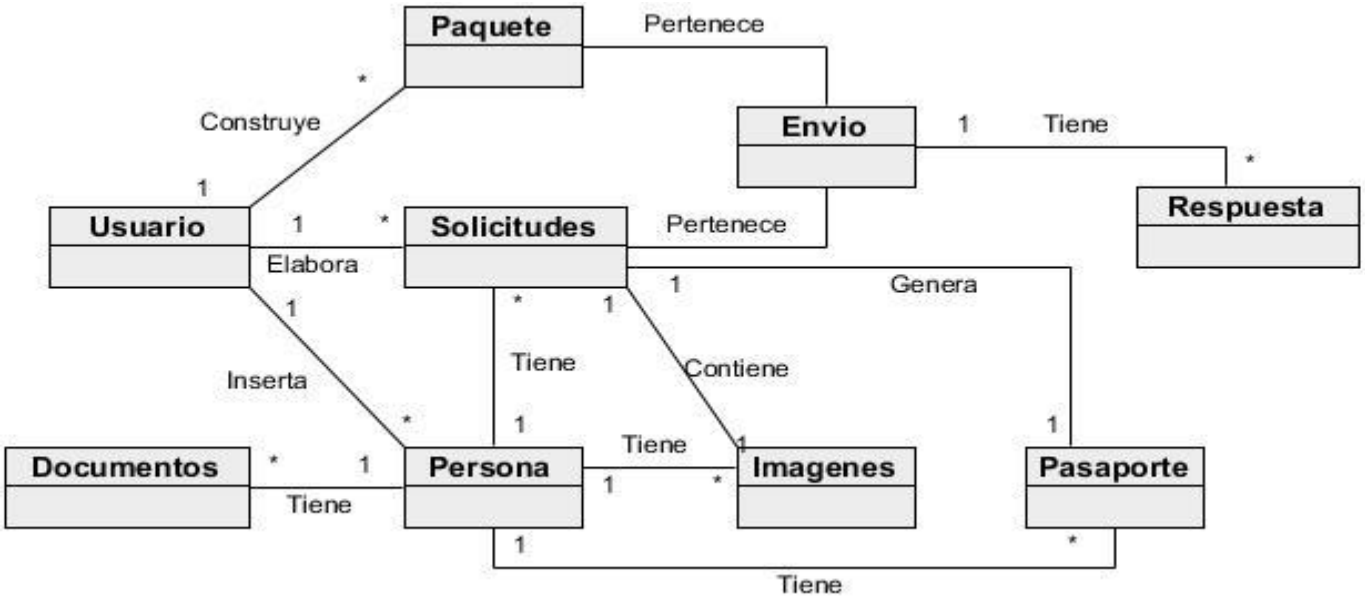

Ilustración 4: Modelo del dominio.

2.2 Modelo del sistema.

El análisis previo de las características del proceso de trámite de pasaporte, necesidad y aspectos de interés facilita la definición de los requerimientos funcionales y no funcionales del sistema y los actores del sistema según establece las Historias de Usuario (HU) que define la metodología seleccionada.

2.2.1 Propuesta de solución.

Para solucionar la problemática planteada se pretende elaborar un sistema informático que permita la gestión de datos para las solicitudes de trámites no presenciales de pasaportes. El objetivo que se persigue con el desarrollo del sistema es lograr suprimir las dependencias existentes en la elaboración manual de las solicitudes de trámites, aportar seguridad en el manejo y traslado de los datos mediante el cifrado de la información que se transfiere en un fichero con formato XML, generado dentro de un paquete con las imágenes de las solicitudes. Con la propuesta de solución se mejorarán los tiempos actuales del proceso. Los algoritmos seleccionados para cifrar la información de las solicitudes de trámites son el AES y el RSA, el primero utilizado para cifrar los datos generando una llave que se protegerá con el segundo algoritmo. La función resumen seleccionada es MD5, utilizada para verificar la integridad de las imágenes asociadas a los datos biométricos de las solicitudes.

2.2.2 Especificación de los requerimientos funcionales.

Los requerimientos funcionales (RF) son declaraciones de lo que debe proveer el sistema, de la manera en que éste debe reaccionar y de cómo se debe comportar. Los RF especifican cómo se comporta el sistema en la entrada y salida de información (28).

RF1. Permitir autenticación.

RF2. Gestionar usuario.

RF 2.1 Insertar usuario.

RF 2.2 Modificar usuario.

RF 2.3 Inhabilitar usuario.

RF 2.4 Buscar usuario.

RF3. Gestionar solicitudes de pasaportes.

RF 3.1 Buscar persona.

RF 3.2 Insertar solicitudes de pasaporte.

RF 3.2.1 Definir tipo de trámite.

RF 3.2.2 Insertar datos biográficos de la persona.

RF 3.2.3 Insertar datos biométricos de la persona.

RF 3.2.4 Insertar documentos de la persona.

RF 3.3 Modificar solicitudes de pasaportes.

RF 3.3.1 Modificar datos biográficos de la persona.

RF 3.3.2 Modificar datos biométricos de la persona.

RF 3.3.3 Modificar documentos de la persona.

RF 4. Gestionar paquete de envío.

RF 4.1 Crear paquete.

RF 4.2 Cifrar paquete.

RF 4.3 Exportar paquete.

RF 5. Gestionar reportes estadísticos.

RF 5.1 Obtener listados estadísticos por solicitudes enviadas.

RF 5.2 Obtener listados estadísticos por rango de fecha.

RF 6. Recepcionar respuesta del envío.

RF 6.1 Importar respuesta.

RF 6.2 Mostrar respuesta.

2.2.3 Descripción de los requerimientos funcionales.

La metodología XP describe los requerimientos funcionales mediante las Historias de Usuario (HU). Este artefacto sirve para describir de manera breve los requerimientos, son dinámicas, flexibles y reemplazables en cualquier fase del desarrollo. A continuación se describe el requisito funcional: Gestionar las solicitudes de pasaportes, como ejemplo de la especificación de requerimientos. La descripción de cada Historia de Usuario se especifica en el Anexo 1.

Historia de Usuario	
Número: HU_3	**Usuario:** desarrollador
Nombre de historia de usuario: Gestionar las solicitudes de pasaportes.	
Prioridad en negocio: alta	**Iteración asignada:** 1
Riesgo en desarrollo: baja	**Puntos estimados:** 3
Programador responsable: Yoelmy Hernández Barzaga	
Descripción: Permite gestionar la información de las solicitudes de pasaportes que se registran.	
Observación: Gestionar las solicitudes de pasaportes incluye buscar persona además de insertar y modificar las solicitudes de pasaportes.	

Tabla 3: HU_3 Gestionar las solicitudes de pasaportes.

2.2.4 Requerimientos no funcionales.

Los requerimientos no funcionales (RNF) son aquellas propiedades o restricciones que posee el sistema, desde el entorno de ejecución o la gama de colores permisible hasta los sistemas operativos donde es aplicable (28). Los RNF también aportan características atractivas, de usabilidad y fiabilidad al producto.

- **Requerimientos del producto.**

  Requerimientos de usabilidad:

  - El sistema debe ser de tipo “Escritorio”.
  - El sistema podrá ser usado por cualquier persona.

  Requerimiento de fiabilidad:

  - El sistema debe estar disponible en cualquier momento.

  Requerimiento de desempeño:

  - El sistema está concebido para entornos donde los tiempos de respuesta deben ser de 3 segundos por cada paquete generado así como el procesamiento de la información.

  Requerimientos de portabilidad:

  - Es compatible con los sistemas operativos Windows XP o superior y GNU/Linux (Ubuntu 9.10 o superior, Solaris OS versión 11, Debian).
  - Se necesita la versión 9.2.4 de PostgreSQL como gestor de base de datos.
  - Se necesita el JDK 7 para ejecutar la aplicación.
  - La aplicación es ejecutable en computadores con un mínimo de 256 MB[32] de memoria RAM[33] , espacio en disco de 750 MB y procesador Intel Pentium III; se recomiendan 512 MB de

[32] *MegaBytes*.
[33] *Random Access Memory* (Memoria de Acceso Aleatorio).

memoria RAM y procesador Intel Pentium IV o superior (requerimientos mínimos y recomendados del lenguaje *Java*).

- **Requerimientos organizacionales.**

  Requerimientos de implementación:

  - El diseño gráfico del sistema no debe ser complejo, con colores suaves que no afecten la visión de los usuarios.

- **Requerimientos externos.**

  Requerimiento de seguridad:

  - El sistema debe contar con un administrador que controle y gestione los usuarios, privilegios y acceso a recursos.
  - Permisos de usuarios limitados por roles.

2.3 Metáfora.

La metáfora describe cómo debe funcionar el sistema. La metodología seleccionada sugiere la utilización de esta definición como una manera simple de explicar el propósito de la solución. Esta metáfora permite elaborar un conjunto de nombres que sirvan como términos que hacen referencia al dominio del problema. El cliente y los desarrolladores deben estar de acuerdo con la metáfora para evitar incoherencias en los términos y lograr un consenso en lo que se pretender implementar (18).

El sistema de gestión de datos para las solicitudes de trámites no presenciales de pasaportes puede ser utilizado en cualquier entidad que necesite realizar dichos trámites, sin la necesidad de conexión. El sistema puede ser accesible para cualquier usuario previamente registrado y asignado a uno de los roles definidos para el manejo de recursos y/o privilegios. Se capturan los datos biográficos, biométricos y los documentos de cada persona, los cuales conforman la solicitud de trámite. Se crean paquetes compuestos por una o varias solicitudes, los cuales se generan en formato XML[34], con la información cifrada, para ser enviados hacia las oficinas expedidoras con mayor seguridad en la información. Se gestionan los usuarios que interactúan con el sistema, siendo posible insertar, modificar, habilitar e inhabilitar los mismos. Se pueden realizar búsquedas por solicitudes enviadas, rango de fechas de las solicitudes, números de identidad, primer nombre, segundo nombre, primer apellido y segundo apellido generando reportes estadísticos según los resultados. Los datos capturados es posible modificarlos mediante el acceso a las búsquedas en el sistema.

[34] *eXtensible Markup Language* (Lenguaje de Marcas Extensibles).

2.4 Diseño de la solución.

2.4.1 Tarjetas CRC.

Las tarjetas CRC (Clase-Responsabilidad-Colaborador) es un artefacto que guía al sistema a través del análisis de las responsabilidades. Posibilita estudiar las clases y refinarlas según su responsabilidad dentro del sistema y las clases con las que colaborar (29). En el diseño de la propuesta de solución se identificaron un conjunto de clases que por su responsabilidad resultan claves en la implementación del sistema en los distintos niveles definidos por la arquitectura seleccionada. Para manejar el acceso a datos las clases DAO e HibernateFactory, para controlar el negocio las clases StnpEncriptarAES y StnpConrolNegocioSolicitud, y para manejar la presentación con el usuario la clase IfrmPrincipal. A continuación se describe la tarjeta CRC de la clase DAO, las demás pueden ser consultadas en el Anexo 3.

DAO	
Responsabilidades	**Colaboradores**
Clase que trabaja en la capa de persistencia de datos, encargada de manejar las sentencias de manipulación de datos (insertar, modificar, actualizar o borrar).	java.util.LinkedList; java.util.List; org.hibernate.HibernateException; org.hibernate.Query; org.hibernate.Session; org.hibernate.Transaction; cu.uci.cised.stnp.acceso_datos.Ncolorcabello cu.uci.cised.stnp.acceso_datos.Ncolorpiel cu.uci.cised.stnp.acceso_datos.Imagenes cu.uci.cised.stnp.acceso_datos.Nmunicipio cu.uci.cised.stnp.acceso_datos.Ntipodocumento cu.uci.cised.stnp.acceso_datos.Ngenero cu.uci.cised.stnp.acceso_datos.Usuario cu.uci.cised.stnp.acceso_datos.Ntipotramite cu.uci.cised.stnp.acceso_datos.Nprofesion cu.uci.cised.stnp.acceso_datos.Nprovincia cu.uci.cised.stnp.acceso_datos.Npais cu.uci.cised.stnp.acceso_datos.Ncolorojos cu.uci.cised.stnp.acceso_datos.Persona cu.uci.cised.stnp.acceso_datos.Documento

Tabla 4: Tarjeta CRC de la clase DAO.

2.4.2 Diagrama de clases.

Los diagramas de clases permiten un mejor entendimiento de las funciones entre archivos y clases desarrollados. A continuación se representa el diagrama de paquetes y el diagrama de clases

correspondientes al sistema de gestión de datos para las solicitudes de trámites no presenciales de pasaportes. El diagrama de paquetes está compuesto por 8 paquetes, de las cuales los paquetes presentación, negocio, acceso_datos, y dao son los principales; en el primero se localizan las clases que permiten la interacción del usuario con el sistema, en el segundo se encuentran las clases donde se implementan los requerimientos del sistema y donde se desarrolla el negocio, en el tercero se localizan la clases que interactúan con las entidades definidas en la base de datos y en el cuarto se encuentran las clases que facilitan la persistencia con los datos almacenados. Las clases asociadas a estos paquetes principales se representan en los diagramas de clases que se muestran a continuación en este epígrafe; la ilustración 9 solo representa algunos de los métodos y atributos de las clases, para mayor información ver Anexo 4.

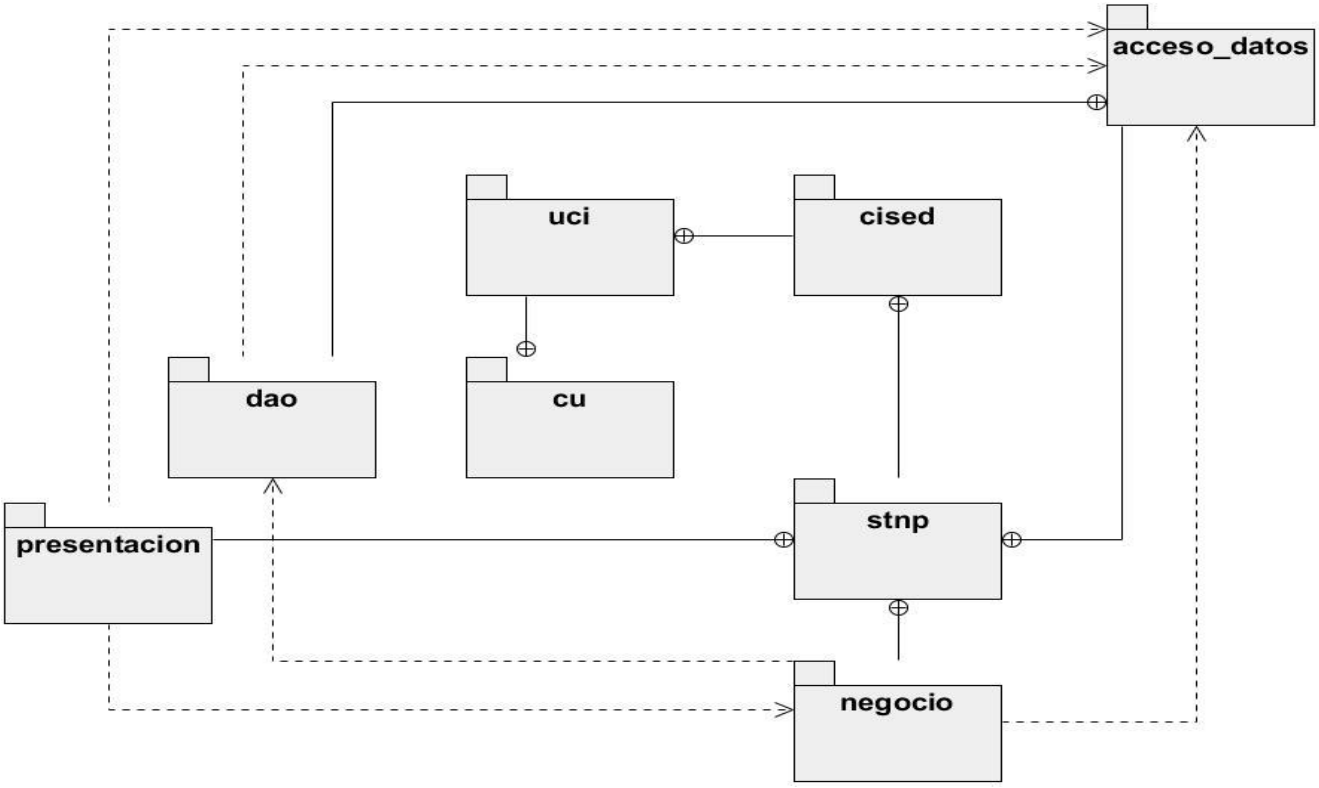

Ilustración 5: Diagrama de paquetes del sistema.

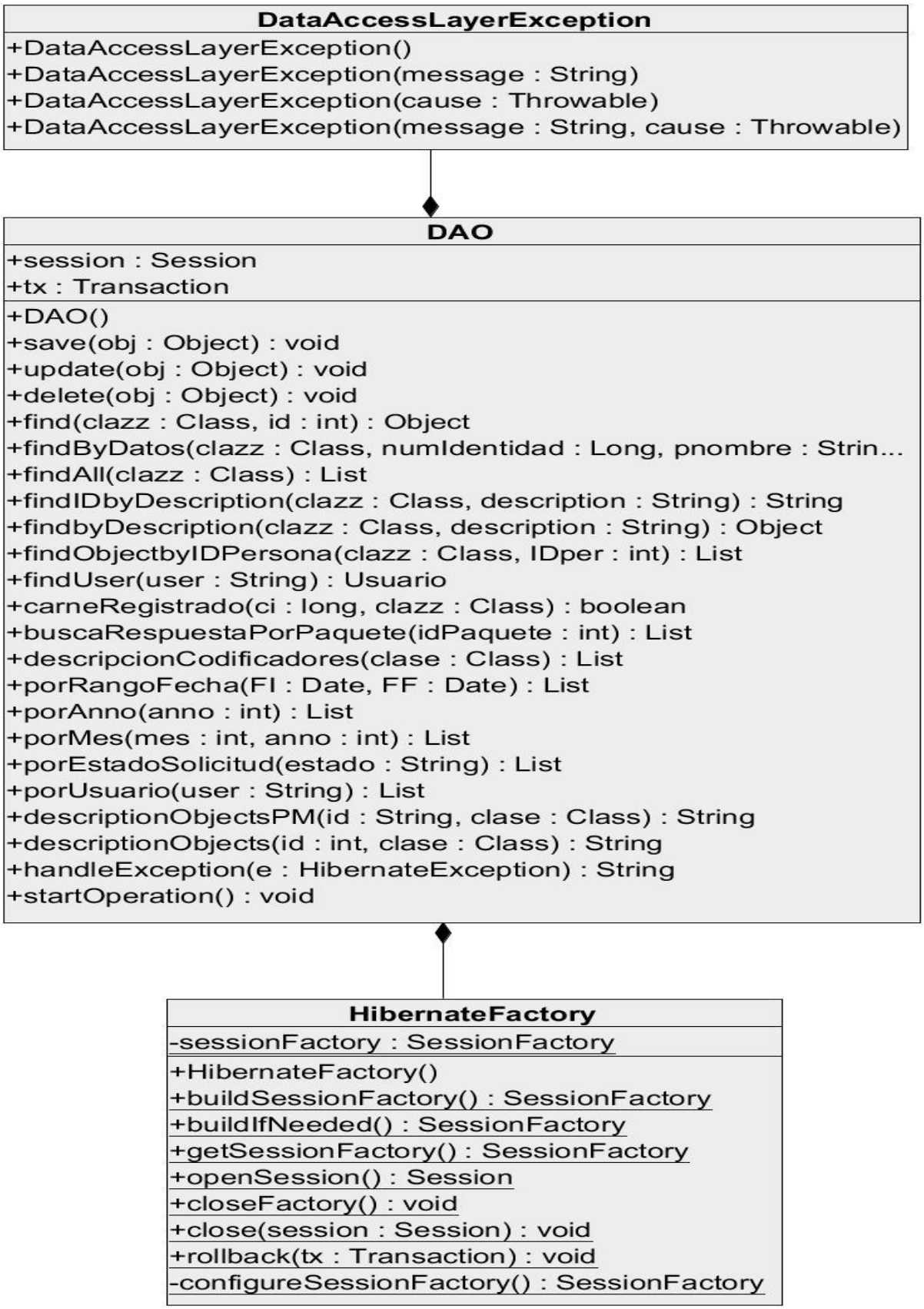

Ilustración 6: Diagrama de clases pertenecientes al paquete dao.

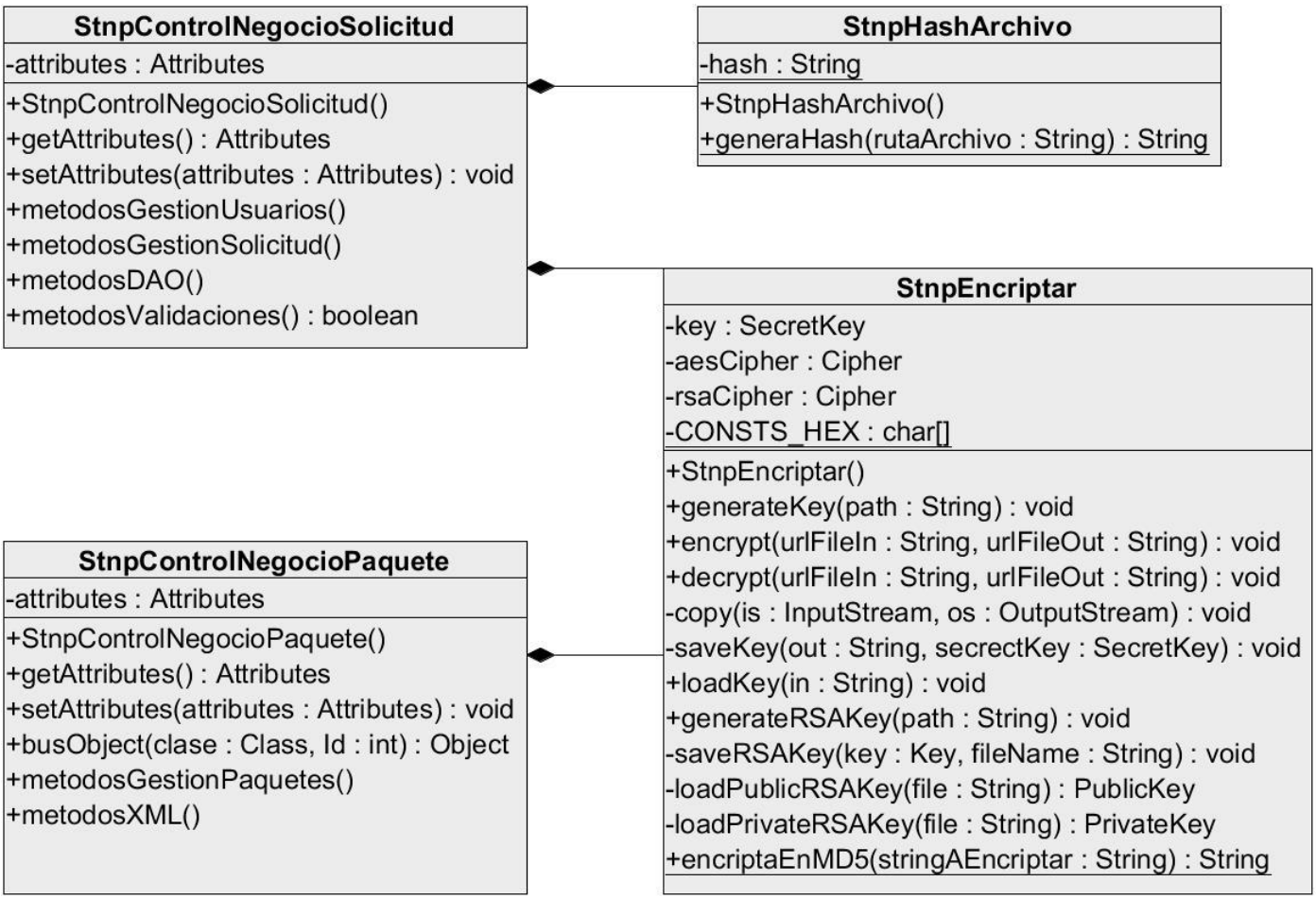

Ilustración 7: Diagrama de clases pertenecientes al paquete negocio.

2.4.3 Patrón arquitectónico.

La ejecución del proceso de implementación del sistema de gestión de datos para la solicitud de trámites no presencial de pasaporte se estructura mediante el uso de la **arquitectura basada en capas**. La arquitectura de software de un programa o sistema de cómputo según aborda Roger S. Pressman: “es la estructura o las estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos (Bass, Clement y Kazman [BAS03])” (30).

La aplicación de esta arquitectura brinda la posibilidad de separar de una forma lógica el proyecto. Los niveles o capas en los que se puede separar la aplicación ofrecen un entendimiento de forma efectiva qué es lo que se necesita y cómo es posible ejecutarlo. El uso de esta arquitectura facilita la ejecución de una aplicación con posibilidad de adaptarse al entorno y de reaccionar ante los posibles cambios durante su empleo en el negocio. Con el uso de este patrón arquitectónico es posible enmendar errores en cualquier nivel sin verse afectado los demás, lo que propicia menos retraso en el proceso de desarrollo (18).

Características de la arquitectura en capas.

- Datos y servicios separados.

- Lógica del negocio y datos separados.
- Separación entre la lógica de la aplicación y la interfaz, ofreciendo al usuario mayores opciones de diseño.
- Brinda apoyo en la división de tareas además de seguridad en el trabajo con las capas.

La arquitectura basada en capas posee varias formas de aplicar su filosofía, ya sea mediante el uso de tres capas, cuatro capas o n capas. Cada uno de estas formas de ejecución varía según las necesidades del negocio. En el desarrollo del sistema de gestión de datos para la solicitud de trámites no presenciales de pasaportes se implementó el uso de tres capas que a continuación se describen:

- **Capa de presentación:** este nivel o capa es la encargada del manejo con la interfaz gráfica del usuario. Se puede visualizar el sistema y acceder a toda la información contenida en la aplicación. En este nivel se trabajan los formularios y componentes del entorno visual del sistema de gestión de datos. De todos los niveles este es el que actúa directamente con el usuario.
- **Capa de negocio:** en esta capa es donde reside toda la lógica del negocio, o sea es el nivel en el que se tratan todos los requerimientos funcionales captados. Esta capa se comunica con la capa de presentación permitiendo que esta última muestre al usuario los datos solicitados. El nivel de la lógica del negocio procesa todas las políticas y normas necesarias para cumplir con los requerimientos del sistema.
- **Capa de acceso a datos:** este nivel manipula todos los datos que son necesarios almacenar, mostrar y modificar. En esta capa se encuentra guardada toda la información de la lógica del negocio y del usuario. La función de esta capa de acceso a datos es la de mantener almacenados todos los datos que son manipulados y mostrar la información solicitada porque es la única capa que tiene acceso a esos datos. Esta capa se comunica con la capa de negocio cuando esta última le solicita guardar o mostrar información de la base de datos.

El uso de la arquitectura basada en capas, y específicamente con el uso de tres capas, brinda múltiples funciones y características afines con las necesidades de implementación. Permite que las capas estén distribuidas de modo tal que solo se comunican la inmediata superior o inferior. Este patrón en tres capas propone una mejor organización del sistema además de un mejor entendimiento del mismo, permitiendo que la independencia entre capas no afecte el proceso lo cual propicia que su ejecución sea factible.

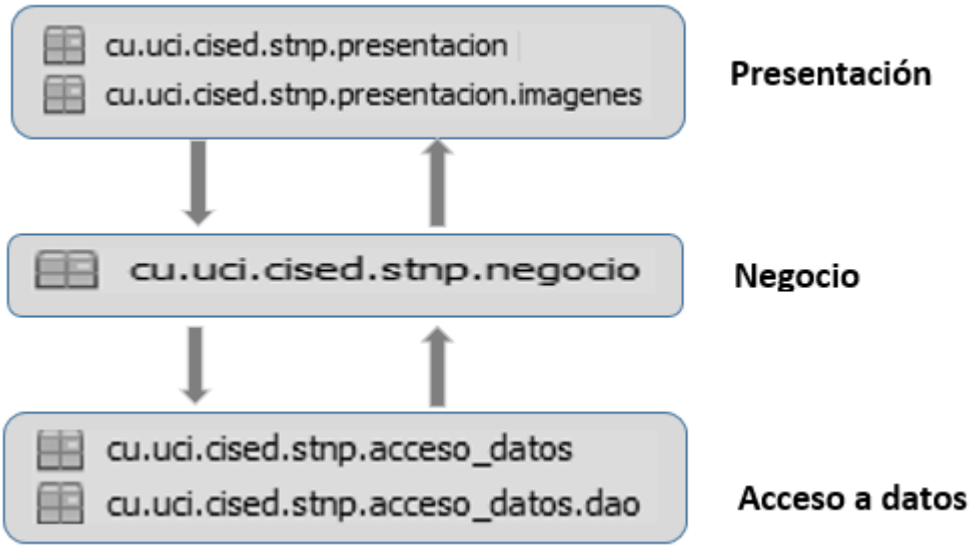

Ilustración 8: Modelo de la arquitectura implementada.

2.4.4 Modelo de datos.

El modelo de datos propuesto representa la realidad del proceso de trámites mediante un esquema gráfico representando las entidades que intervienen. Estas entidades son identificadas en el problema a resolver y sus rasgos distintivos se denominan atributos. A continuación se representa el diagrama entidad-relación, pero solo con el nombre de las entidades que intervienen en el proceso, para mayor información consultar el Anexo 5.

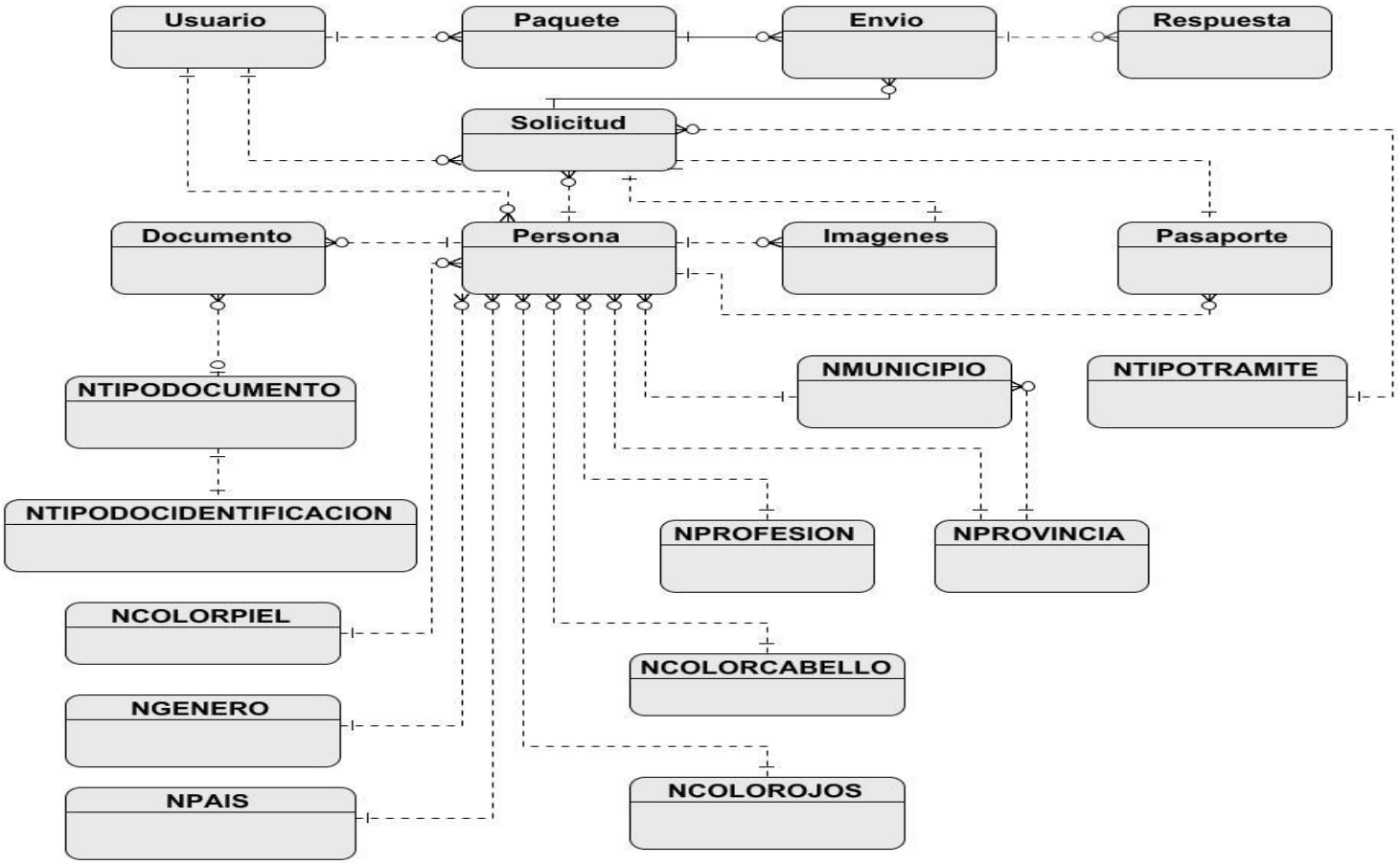

Ilustración 9: Diagrama entidad-relación.

2.4.5 Patrones de diseño.

En la terminología de objetos, el patrón: "es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas" (29).

Existen diversos patrones que definen las responsabilidades asignadas a los distintos objetos. En la ejecución de la propuesta de solución se utilizó el patrón GRASP[35] el cual "describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones" (29); y los patrones GOF[36] que describen 23 patrones de diseños descritos en el libro "*Design Patterns: Elements of Reusable Object Oriented Software*", los cuales se encuentran divididos en tres categorías según su

[35] *General Responsibility Assigment* Software *Patterns* (Patrones de Principios Generales para Asignar Responsabilidades).
[36] *Gang of Four* (Pandilla de los Cuatro).

objetivo: Creacionales, Estructurales y de Comportamiento y dirigen el desarrollo de aplicaciones orientados a objetos .

2.4.5.1 Patrones GRASP.

Los patrones GRASP que se utilizan en la propuesta de solución se describen a continuación (29).

- **Experto:** soluciona el problema de asignar una responsabilidad al experto en la información, es decir a la clases donde se encuentra la información necesaria. Posibilita originar un diseño en el cual el objeto responsable realiza las operaciones que realmente representa. Este patrón facilita la asignación de las responsabilidades a las clases según les corresponda. Mantiene el encapsulamiento debido a que los objetos se sirven de sus propios datos para ejecutar órdenes, soportando bajo acoplamiento y generando un sistema resistente y factible.
  El uso de este patrón se evidencia en la clase StnpEncriptarAES.java.
- **Creador:** Posibilita dar solución al problema de quién debería ser el responsable de crear nuevas instancias de una clase. La creación de instancia es una actividad cotidiana en la programación orientada a objetos por lo que es útil contar con un patrón para asignar responsabilidades de este tipo. La adecuada aplicación de este patrón permite soportar bajo acoplamiento, encapsulación y reutilización.
  Ejemplo del uso del patrón se observa en la clase StnpControlNegocioSolicitud.java.
- **Bajo acoplamiento:** Brinda la posibilidad de asignar una responsabilidad para mantener bajo acoplamiento. Mediante este patrón es posible tener una medida de la conexión existente entre las clases y el hecho de tener un bajo acoplamiento permite concluir que no existe una alta dependencia con las demás. Este patrón ofrece un soporte de diseño de forma tal que existan clases independientes a fin de reducir el impacto que puedan producir los cambios. Posibilita que no se afecten por cambios en los componentes, facilidad de entendimiento y de reutilización.
  Ejemplo del uso de este patrón se evidencia en la clase StnpControlNegocioPaquete.
- **Alta cohesión:** Soluciona el problema de asignar una responsabilidad de manera tal que la cohesión se mantenga alta. Desde el punto de vista del diseño brinda una métrica de la relación y el enfoque que poseen las responsabilidades de una clase. La alta cohesión de las clases caracteriza a las clases con responsabilidades que no realicen un trabajo excesivo. En cuestiones de diseño este principio se debe tener presente debido a que plantea una meta para los desarrolladores. Mejora la claridad y facilidad del diseño, simplicidad en el mantenimiento y mayor capacidad de reutilización.
  Ejemplo del uso de este patrón se evidencia en la clase StnpHashArchivo.
- **Controlador:** Contribuye a solucionar el problema de recibir y manipular mensajes del sistema. Resuelve el dilema de quién debe ser el responsable de gestionar los eventos de entrada al sistema.
  Ejemplo del uso de este patrón se observa en la clase StnpControlNegocioSolicitud.

2.4.5.2 Patrones GOF.

Los patrones GOF que se utilizan en la propuesta de solución se describen a continuación (31).

- **Creacionales:** manejan el tratamiento de objetos debido a que posibilitan la abstracción del proceso de instanciación y no mostrar aspectos de cómo se construye dicho objeto.
  - ***Builder* (Constructor):** separa la construcción y representación de un objeto, centralizando este proceso y dando la posibilidad que sirva para representar distintas instancias. El uso de este patrón brinda flexibilidad al diseño, la independencia posible entre la construcción y representación promueve la reutilización y se favorece el encapsulamiento y control.

    Este patrón se evidencia en la clase Solicitud.java.
  - ***Singleton* (Instancia única):** garantiza una instancia única para una clase y el mecanismo para acceder de forma global a dicha clase. El acceso a esta instancia está controlado, posibilita refinar operaciones y representación, se reduce el espacio de nombres y es más flexible que otras alternativas.

    Este patrón se utiliza en la clase HibernateFactory.java.
- **Comportamiento:** ayudan a establecer la comunicación e iteración entre los objetos del sistema. Reduce el acoplamiento entre objetos.
  - ***Observer* (Observador):** define la dependencia de uno-a-muchos entre las los objetos de forma tal que si uno cambia de estado sus dependencias son advertidas y puedan actualizarse. Posibilita la existencia de variación y evolución en ciertos detalles de forma independiente, facilita las notificaciones ya que solo se comunican y las dependencias actúan en consecuencia.

2.5 Estimación del esfuerzo.

La elaboración de las historias de usuario conlleva un esfuerzo en su ejecución. Los desarrolladores estiman el tiempo de esfuerzo que conlleva la implementación de cada HU. Este valor se define en semanas y no es exacto en su totalidad.

Historia de usuario	Estimación (semanas)
Permitir autenticación	1
Gestionar usuario.	2
Gestionar las solicitudes de pasaportes.	3
Gestionar paquete de envío.	3
Gestionar reportes estadísticos.	1
Recepcionar respuesta de envío.	1

Tabla 5: Estimación de tiempo por Historia de Usuario.

2.6 Plan de iteraciones.

La metodología seleccionada propone la creación del plan de duración de las iteraciones definidas, permitiendo observar el orden de implementación y cuánto durará la ejecución de las iteraciones (medidas en semanas). Se tienen definidas 6 HU, con los requerimientos funcionales implícitos en cada una y estas se desarrollarán en 3 iteraciones abarcando un total de 11 semanas. En el Anexo 6 se describe con más detalles el plan propuesto.

2.7 Plan de entregas.

Al concluir la elaboración de las HU se procede a la elaboración del plan de entregas mediante el cual se estima el tiempo de desarrollo. Este artefacto determina el periodo de tiempo que demora la implementación de las HU, con el mismo quedan definidas las fechas de entrega de las versiones funcionales. En el Anexo 7 se describe el plan de entregas de la propuesta de solución.

2.8 Conclusiones del capítulo.

Después de realizar el análisis y diseño de la propuesta de solución y determinar sus características teniendo como referente el estudio de los requerimientos que debe tener el sistema y el análisis del negocio, se logra un mejor entendimiento del diseño de la solución y del flujo de procesos que se desea informatizar.

- Se definieron las HU, y en consecuencia, el plan de duración y entrega para el desarrollo del sistema, logrando una mejor visión de la arquitectura a implementar.
- El uso apropiado de los patrones de diseño seleccionados permitirá construir un sistema óptimo para el desarrollo del flujo de eventos definidos.

Capítulo 3. Implementación y prueba.

Después de concluir el diseño de la propuesta de solución y definir la arquitectura a implementar, se ejecuta la construcción del sistema. En el presente capítulo se describe todo el proceso de desarrollo de la propuesta de solución. Se describen los artefactos del desarrollo, diagrama de componentes y las distintas pruebas a realizar.

3.1 Estándar de codificación.

En el ámbito de la programación es necesario establecer un criterio que acepte normas en la implementación del código fuente de la propuesta de desarrollo. Definir un estándar facilita el entendimiento de lo que se construye puesto que establece normas a seguir por los programadores. Para el lenguaje definido se utilizan estándares de codificación recomendados por sus desarrolladores, permitiendo que exista facilidad en el entendimiento, manejabilidad y legibilidad del código.

3.1.1Estándar de programación.

3.1.1.1 Organización de ficheros.

En el lenguaje Java, las clases se encuentran agrupadas en paquetes. Estas estructuras presentan una organización jerárquica y es posible crear sub-paquetes dentro de las mismas. Los ficheros tienen secciones separadas por líneas en blanco y se debe evitar que excedan las mil líneas de código.

- **Ficheros fuentes:** cada uno de estos ficheros (.java) debe presentar una única interfaz y se distinguen los comentarios de inicio, sentencia de paquete, sentencias de importación y declaración de clases e interfaces.

Ejemplo:"cu.uci.cised.stnp.accesodatos", de forma que en ese paquete se encuentra el código referente al acceso a datos del proyecto STNP[37] desarrollado para el centro CISED[38] perteneciente a la UCI, Cuba.

3.1.1.2 Comentarios.

Los comentarios de implementación se utilizan para describir información relacionada con la implementación del código. Se describen aspectos como descripción de variables, fases y excepciones. Se manejan los comentarios de bloque, de líneas y de fin de línea.

Ejemplos:

- Comentario de bloque: **/***
  *** Comentario**
  ***/**
- Comentario de línea: **// Comentario**
- Comentario a final de línea: **int a = 2 + 4; // Inicialización de la variable.**

[37] Sistema de Trámite No Presencial.
[38] Centro de Identificación y Seguridad Digital.

3.1.1.3 Nomenclatura de identificadores.

Se establecen identificadores para los paquetes, clases, métodos, variables y constantes utilizadas con el objetivo de proporcionar una solución comprensible. Los paquetes serán escritos con letras minúsculas, el prefijo será en correspondencia con el dominio de primer nivel establecido como: cu, ar, es, org, com o net; lo demás que conforma el nombre del paquete se establecerá según la organización, proyecto, sección u otros esquemas.

Las clases deben nombrarse con letra inicial mayúscula, en caso de ser compuesto cada palabra que forma el nombre debe empezar con mayúscula. Se deben evitar el uso de abreviaturas o acrónimos. Los métodos deben ser verbos escritos en minúscula y en caso de ser compuesto la segunda palabra con letra inicial mayúscula. Las variables serán escritas con letras minúsculas y si es compuesta, la segunda palabra inicial con letra mayúscula, no deben comenzar con caracteres tales como "_" o "$" y deben expresar con claridad el significado de su función. Las constantes se deben escribir con letras mayúsculas y de ser compuestos serán separadas por el carácter "_".

Ejemplos:

- Clases: **class Persona class Solicitud**
- Métodos: **public int usuarioValido ();**
- Variables: **Persona persona Solicitud solicitud**
- Constantes: **int LONGITUD_LISTA int CONTADOR**

3.1.1.4 Prácticas de programación.

La visibilidad de atributos de instancia y clase y la asignación sobre las variables también son un aspecto a tener presente en la ejecución del código. La longitud de las líneas no debe exceder de 80 caracteres debido a que dificulta la visualización y lectura de datos. Los atributos serán siempre privados, excepto que sea necesario su uso por subclases herederas y se declara como protegidos, la utilización de estos atributos será mediante los "get" y "set". Se debe evitar la asignación de un mismo valor respecto a diferentes variables dentro de una misma línea de código debido a que estas sentencias pueden ser difíciles de leer, no es recomendable la utilización de operadores de asignación donde pueda existir confusión con el operador de igualdad.

Ejemplo:

- Visibilidad de atributos: **public class Persona;**
  private long numeroIdentidad;
- Asignación: **int N = M = Z = 10;** → Evitar esta práctica.

if ((N = M++) == 0) → Evitar esta práctica.

3.2 Tareas de ingenierías.

La metodología seleccionada establece las tareas de ingenierías, donde se describen las tareas que le son asignadas al programador. Este artefacto, que establece la *eXtreme Programming*, facilita la

implementación de las HU definidas durante la iteración que le corresponda. A continuación se describe las tareas de ingenierías correspondientes a las HU de la primera iteración, las restantes pueden ser consultadas en el Anexo 8.

Iteración 1	
Historias de usuario	**Tareas**
Permitir autenticación	• Validar campos de entrada de datos. • Cifrar contraseña del usuario. • Verificar los parámetros de entrada. • Ejecución de un mensaje de texto informando si los el usuario y/o la contraseña son incorrectos, sino ingresa al sistema.
Gestionar usuario	• Verificar campos de entrada de datos. • Verificar la selección del rol. • Cifrar contraseña del nuevo usuario. • Verificar que el usuario no exista. • Ejecución de un mensaje de texto informando que se guardó satisfactoriamente el usuario.

Gestionar las solicitudes de pasaporte	• Buscar persona.
	• Validar campos de entrada de datos biográficos y biométricos.
	• Insertar solicitudes de pasaporte.
	  ○ Definir tipo de trámite.
	  ○ Insertar datos biográficos de la persona.
	  ○ Insertar datos biométricos de la persona
	• Ejecución de un mensaje de texto informado que se guardó satisfactoriamente la solicitud.

Gestionar paquete de envío	• Mostrar solicitudes según criterio de búsqueda. • Listar las solicitudes encontradas que conformarán el paquete. • Generar fichero XML con las solicitudes seleccionadas. • Crear paquete.
Gestionar reportes estadísticos	• Mostrar solicitudes según criterio de búsqueda: solicitudes enviadas. ○ Se listan las solicitudes que coinciden con el criterio de búsqueda. • Ejecución de un mensaje de texto informando que la búsqueda produjo resultados o que no se encontraron solicitudes asociadas.

Tabla 6: Tareas de ingeniería en la primera iteración.

3.3 Tratamiento de errores.

Los programas suelen contener disímiles inconvenientes que pueden afectar el flujo normal del proceso de construcción del software y su posterior utilización por el usuario final. En el desarrollo de aplicaciones, es

posible la existencia de errores en la implementación, por lo que el tratamiento que se le dé a estas potenciales fallas es de vital importancia en el proceso de construcción.

Para solucionar estos inconvenientes es posible aplicar diversas variantes, ya sea utilizando técnicas implementadas por los propios desarrolladores o mediante la utilización de los recursos que brindan los lenguajes de programación para el tratamiento de errores. El lenguaje *Java* gestiona los errores y situaciones excepcionales como parte de su mecanismo, y es comúnmente conocido como tratamiento de excepciones. En el desarrollo de la propuesta de solución se definió que es factible el tratamiento de excepciones haciendo uso de la gestión de errores que implementa el lenguaje seleccionado debido a que el mismo posee una estructura bien definida y probada para el manejo de errores. Las excepciones en *Java* tratan con una amplia variedad de casos especiales, separan el manejo de errores del código propio, además de agrupar los tipos de errores y diferenciarlos.

Las excepciones se definen como: “cierto tipo de error o una condición anormal que se ha producido durante la ejecución de un programa.” (32), pueden dividirse en dos tipos: las verificadas (*checked*) y las no verificadas (*unchecked*). Las excepciones verificadas hacen referencia a los errores de los cuales es posible recuperarse, se implementan requerimientos para atraparlas o declararlas y corregirlas. Estas excepciones se capturan o relanzan mediante el uso del bloque try-catch, el cual trata con códigos que puedan propiciar algún error y capturar el posible fallo.

```
try {
    FileInputStream myStream = new FileInputStream(file);
    for (int readNum; (readNum = myStream.read(bytes)) != -1;) {
        bos.write(bytes, 0, readNum);
    }
} catch (IOException e) {
    Logger.getLogger(ByteArrayOutputStream.class.getName()).log(Level.SEVERE, null, e);
}
```

Ilustración 10: Ejemplo de bloque try-catch utilizado en el tratamiento de errores.

Las excepciones no verificadas representan un error de programación e indican que se deben atrapar y declarar. Estas excepciones no se fuerzan en su captura, o sea no se comprueban por lo que no se utilizan los bloques try-catch, pero es posible que afecte a métodos o clases cuando se necesiten.

En *Java* todas las excepciones heredan de la clase *Throwable,* de la que se derivan las clases Error, que representa los errores que una aplicación no debería intentar trabajar con ellos (errores con la máquina virtual de *Java*), y *Exception* que trata aquellos errores que se gestionan con más frecuencia.

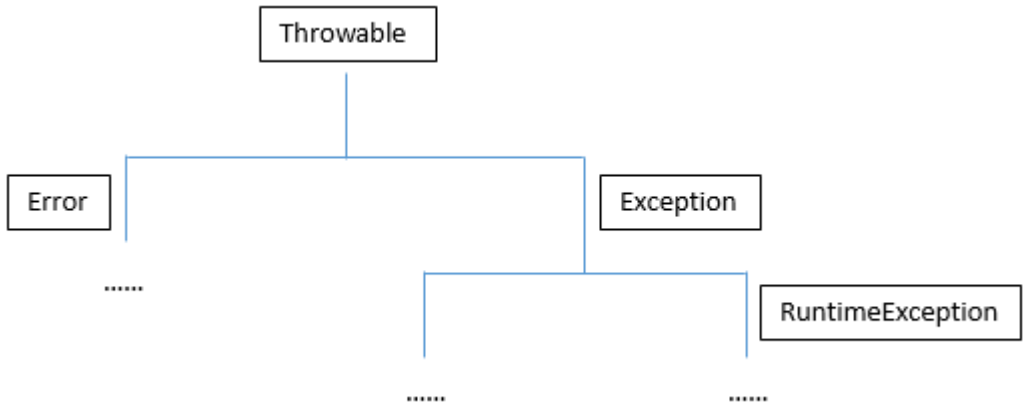

Ilustración 11: Árbol de herencia de las excepciones en *Java*.

En el tratamiento de los errores en la ejecución del acceso a los datos además del trabajo con las excepciones propias del lenguaje se utilizó el mecanismo de gestión de errores que posee la herramienta usada para el mapeo objeto-relacional con la base de datos. *Hibernate* implementa la clase *HibernateException* que hereda de la clase *RuntimeException* y maneja la mayor parte de los posibles fallos en la capa de persistencia y también se hace uso de la clase *DataAccessLayerException*, que posibilita manejar otros errores en la capa de acceso a datos. Para lograr un mejor tratamiento de excepciones en esta capa de persistencia si la *session* lanza algún error se deshace la transacción con la llamada a *close*() y se cierra la *session* en un bloque *finally*.

```
try {
    startOperation();
    session.persist(obj);
    tx.commit();
} catch (HibernateException e) {
    handleException(e); //Método que utiliza la clase DataAccessLayerException.
} finally {
    HibernateFactory.close(session);
}
```

Ilustración 12: Ejemplo de uso de la clase HibernateException y DataAccessLayerException dentro de un bloque try-catch-finally.

3.4 Diagrama de componentes.

El diagrama de componentes modela la vista del sistema en cuanto a la relación y funcionamiento entre archivos (18). Se muestran los componentes que conforman el sistema (bibliotecas, archivos y ejecutables) su relación, dependencia y comunicación.

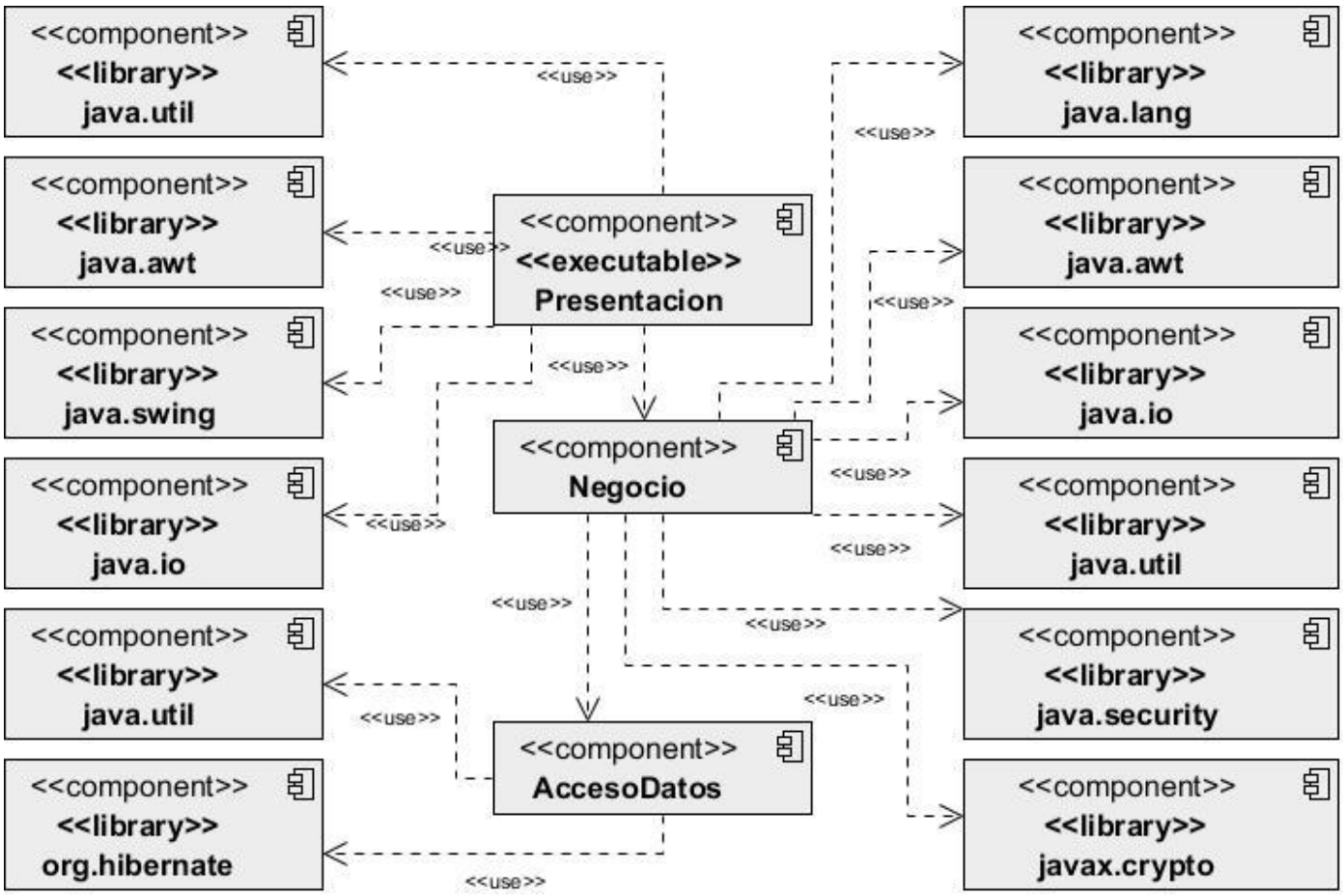

Ilustración 13: Diagrama de componentes.

Descripción de los componentes.

- **Presentación:** contienen todo los elementos de la interfaz gráfica que interactúa directamente con el usuario. Captura los datos y mantiene comunicación con el componente negocio.
- **java.util:** componente librería que accede a los recursos del sistema mediante clases.
- **java.awt**: componente librería que contiene clases para crear interfaces de usuario con ventanas.
- **java.swing:** componente librería que contiene clases para crear interfaces.
- **java.io:** componente librería que contiene clases que maneja la entrada/salida.
- **Negocio:** contiene todos los requerimientos definidos, mantiene comunicación con el componente Presentación para procesar la información que se captura y con el AccesoDatos para consultar la información solicitada.
- **java.security:** componente librería que trabaja con clases e interfaces para el marco de la seguridad, uso de claves privadas y públicas.
- **javax.crypto:** componente librería que posibilita el trabajo con clases e interfaces para su aplicación en el cifrado y descifrado.

- **java.lang:** componente librería que proporciona clases para el trabajo con el diseño del lenguaje *Java*.
- **AccesoDatos:** contiene todos los datos e información del sistema, es el encargado de acceder a estos cuando el componente Negocio solicita consultar esos datos.
- **org.hibernate:** componente librería que permite el mapeo objeto-relacional con la base de datos

3.5 Diagrama de despliegue.

El diagrama de despliegue describe la distribución física que tendrá el sistema en el entorno de ejecución (18).

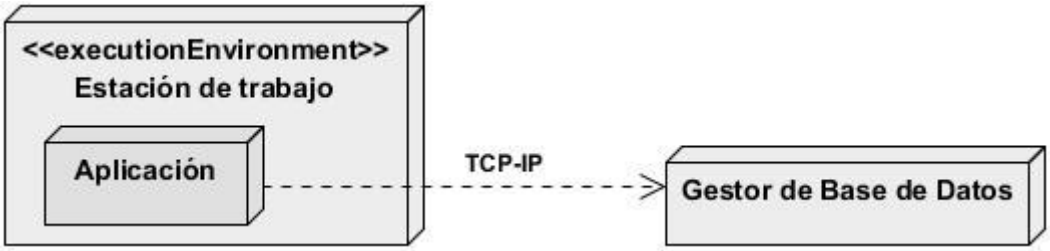

Ilustración 14: Diagrama de despliegue del sistema.

3.6 Pruebas.

En el desarrollo de software la presencia de errores y defectos, en el programa deseado, es común por lo que se hace necesario diseñar y aplicar casos de pruebas al sistema con el fin de verificar si existen defectos. Estas pruebas son procesos que se ejecutan para descubrir la ocurrencia de errores y el mejor de los casos es cuando se detectan la mayor cantidad de problemas posibles especialmente si no habían sido encontrados. Los casos de pruebas tienen como objetivo revelar todos los fallos y errores que puedan existir pero no pueden demostrar que no existan solo que están presentes (30).

Características de los casos de pruebas (30)**.**

- Facilita la búsqueda de la mayor cantidad de posibles errores en la aplicación.
- Están centradas en probar si el sistema no ejecuta lo que debe hacer y si hace lo que no debe hacer.
- No deben ser redundantes ya que el tiempo y recursos disponibles son finitos.
- La mejor prueba es la que tiene una alta probabilidad de encontrar errores.
- No deben ser ni sencillas ni complejas, lo ideal es una combinación de pruebas.

Ilustración 15: Contexto del proceso de pruebas *(34)*.

El proceso de pruebas consta de dos entradas, la primera es la configuración del software que incluye la especificación de los requerimientos, del diseño y del código fuente y la segunda es la configuración de la prueba que contiene un plan y procedimiento de prueba. La correcta ejecución del software y el hecho de que los errores encontrados no sean difíciles de corregir permiten concluir que el software tiene una fiabilidad y calidad aceptables o que los casos de pruebas implementados no son apropiados.

En el entorno de los casos de pruebas se encuentran dos tipos que son aplicados en todos los entornos y niveles: técnicas de pruebas de caja blanca y técnicas de prueba de caja negra. Las técnicas de caja blanca realizan pruebas meticulosas al código del sistema para verificar la lógica interna del programa mientras que las de caja negra se centran en probar la interfaz, las entradas y salidas del mismo.

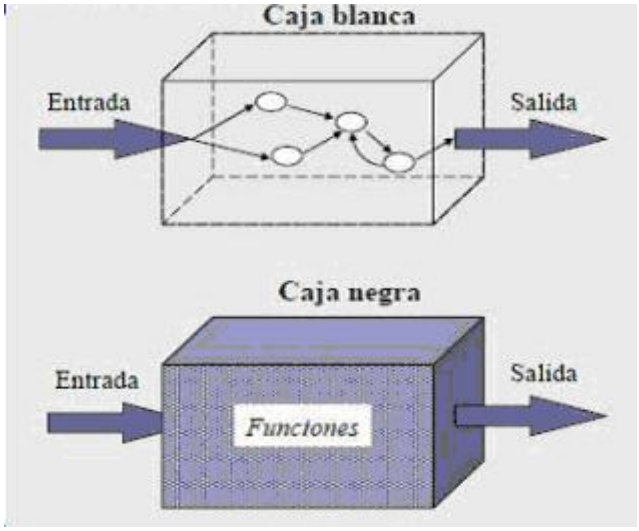

Ilustración 16: Técnicas de pruebas de caja blanca y caja negra *(34)*.

3.6.1 Técnicas de pruebas de caja blanca.

Las pruebas de caja blanca o estructurales se centran en aplicar casos de prueba para verificar el comportamiento del código interno y la estructura del programa. Permite diseñar casos que ejecuten el menos una vez todas las sentencias y condiciones, se verifican todas las opciones tanto verdaderas como falsas, se ejecutan los ciclos hasta el límite y con sus límites operacionales, y son utilizadas las estructuras de datos internas para asegurar validez (30).

Por qué aplicar las pruebas de caja blanca (30)**:**

- Los errores lógicos e incorrectas suposiciones son inversamente proporcional a la ejecución de un camino del programa.
- Es común pensar que un camino lógica posee pocas posibilidades de ejecutarse cuando realmente lo hace de forma normal.
- Los errores tipográficos son aleatorios.

3.6.1.1 Pruebas unitarias.

"La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software: el componente software o módulo" (30). Estas pruebas se estructuran como parte de las pruebas a la interfaz del módulo para garantizar el flujo normal desde y hacia la unidad del programa, se examinan los datos locales para asegurar que los datos mantienen su integridad, se prueban las condiciones límite para comprobar que funcionan en los extremos definidos como restricciones de procesamiento, se recorren todos los caminos independientes para comprobar que las sentencias se ejecutan por lo menos una vez y se prueban los caminos de manejo de errores. Las pruebas unitarias detectan cualquier error en los datos y la lógica de los algoritmos (30).

Para la ejecución de las pruebas unitarias en la propuesta de solución se utiliza el mecanismo de pruebas que posee el lenguaje *Java*. Estas pruebas se realizan a través del *framework* JUnit el cual facilita la aplicación de pruebas unitarias a la aplicación de manera controlada y así poder evaluar el funcionamiento de los métodos de las clases. JUnit verifica, en función de un valor entrado evaluar el valor esperado si esa clase cumple con las especificaciones sino devolverá que esa clase tuvo un fallo en el método correspondiente. A continuación se muestra un ejemplo de la aplicación de las pruebas unitarias realizadas a la funcionalidad usuarioValido() y los resultados obtenidos. Las restantes pruebas unitarias podrán ser consultadas en el Anexo 9.

```java
/**
 * Test of usuarioValido method, of class StnpControlNegocioSolicitud.
 */
@Test
public void testUsuarioValido() {
    System.out.println("usuarioValido");
    String user = "yhbarzaga";
    String pass = "yhbarzaga";
    StnpControlNegocioSolicitud instance = new StnpControlNegocioSolicitud();
    Boolean expResult = true;
    Boolean result = instance.usuarioValido(user, pass);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
}
```

Ilustración 17: Prueba unitaria realizada a la funcionalidad UsuarioValido.

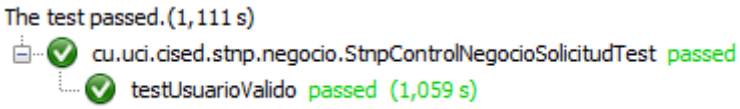

Ilustración 18: Resultado de la prueba unitaria realizada a la funcionalidad UsuarioValido.

3.6.2 Técnicas de pruebas de caja negra.

Las pruebas de caja negra o funcionales, están enfocadas en la verificación de los requerimientos funcionales. Estas pruebas facilitan la obtención de un conjunto de entradas que logren ejecutar todos los requerimientos funcionales con el objetivo de encontrar funciones incorrectas o ausentes, errores de interfaz, errores en las estructuras de datos o en acceso a base de datos y errores de inicialización y de terminación. Las pruebas de caja negra no pretenden ser una variante a las de caja blanca, sino de ser un enfoque agregado para intentar descubrir diferentes tipos de errores que las pruebas de caja blanca (30).

Las pruebas de caja negra permiten responder las siguientes interrogantes (30)**:**

- ¿Cómo se prueba la validez funcional?
- ¿Cómo se prueba el rendimiento y el comportamiento del sistema?
- ¿Qué clases de entrada compondrán unos buenos casos de pruebas?
- ¿Es el sistema particularmente sensible a ciertos valores de entrada?
- ¿Qué volúmenes y niveles de datos tolerará el sistema?

3.6.2.1 Pruebas de aceptación.

Las pruebas de aceptación se crean en correspondencia con las Historias de Usuario, donde se definen varios escenarios para comprobar que cada Historia de Usuario se implementó correctamente. Los clientes son los encargados de verificar que los resultados sean los esperados. A continuación se describe un caso de prueba, en el Anexo 10 pueden ser consultadas las restantes.

Casos de Prueba de Aceptación	
Código de casos de prueba: HU_1_CP1	**Nombre de la Historia de Usuario:** Permitir autenticación
Responsable de la prueba: Yoelmy Hernández Barzaga.	
Descripción de la prueba: Prueba de funcionalidad para autenticación en el sistema. Dado los datos personales del usuario (usuario y contraseña) se verifican para permitir el ingreso al sistema.	
Condiciones de ejecución: El usuario debe existir en el sistema.	
Entrada/Pasos de ejecución: 1. Se introduce usuario y contraseña. 2. Se verifican los datos.	
Resultado esperado: Si los datos insertados son correctos o incorrectos el sistema informa con la ejecución de un mensaje de texto.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 7: HU_1_CP1 Gestionar usuario.

3.6.3 Análisis de los resultados.

Los distintos casos de prueba para los dos tipos de pruebas realizados se aplicaron al sistema en las 4 iteraciones. En cada iteración se obtuvieron no conformidades asociadas a la aplicación de las pruebas descritas, esta información permitió un análisis para evaluar el comportamiento del sistema.

3.6.3.1 Análisis de los resultados de las pruebas de caja blanca.

Las pruebas unitarias fueron aplicadas en 3 iteraciones, según el plan de iteraciones definido para el desarrollo de la propuesta de solución. En la primera iteración, posterior a la aplicación de las Historias de Usuario definidas para esta iteración, se detectaron 9 no conformidades, las cuales fueron rectificadas durante la próxima iteración. En la segunda iteración, posterior a la aplicación de las Historias de Usuario definidas para esta iteración y rectificadas las no conformidades detectadas en la iteración anterior, se revelaron 2 no conformidades que fueron corregidas en la siguiente iteración. En la tercera iteración, posterior a la aplicación de las Historias de Usuario definidas para la misma y rectificadas las no conformidades detectadas en la iteración anterior, no se mostraron inconvenientes.

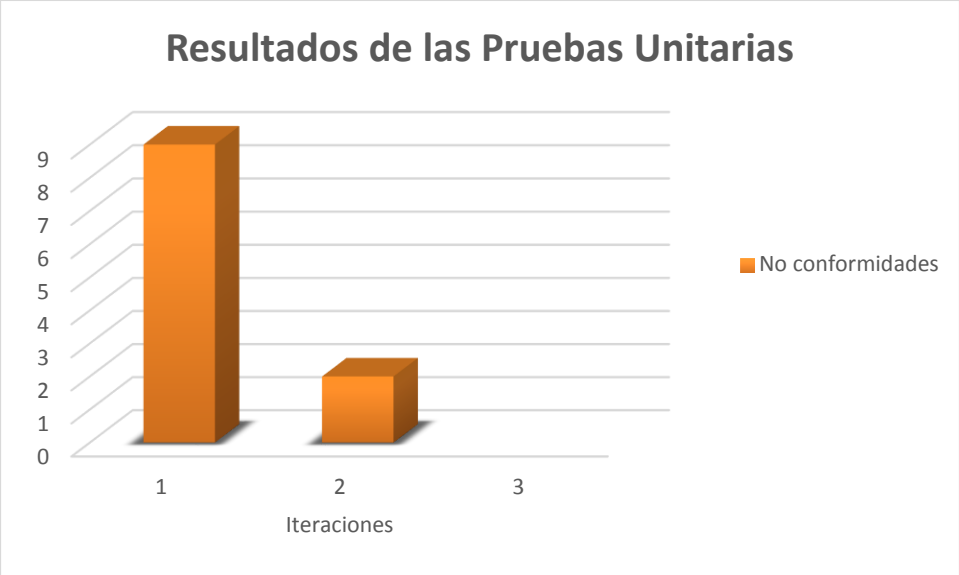

Ilustración 19: Resultados de las pruebas unitarias.

3.6.3.2 Análisis de los resultados de las pruebas de caja negra.

Las pruebas de aceptación fueron aplicadas en 3 iteraciones, según el plan definido para el desarrollo de la propuesta de solución. En la primera iteración, posterior a la aplicación de las Historias de Usuario definidas para esta iteración, se detectaron 15 no conformidades, las cuales fueron rectificadas durante la próxima iteración. En la segunda iteración, posterior a la aplicación de las Historias de Usuario definidas para esta iteración y rectificadas las no conformidades detectadas en la iteración anterior, se revelaron 6 no conformidades que fueron corregidas en la siguiente iteración. En la tercera iteración, posterior a la aplicación de las Historias de Usuario definidas para la misma y rectificadas las no conformidades detectadas en la iteración anterior, no se detectaron inconvenientes.

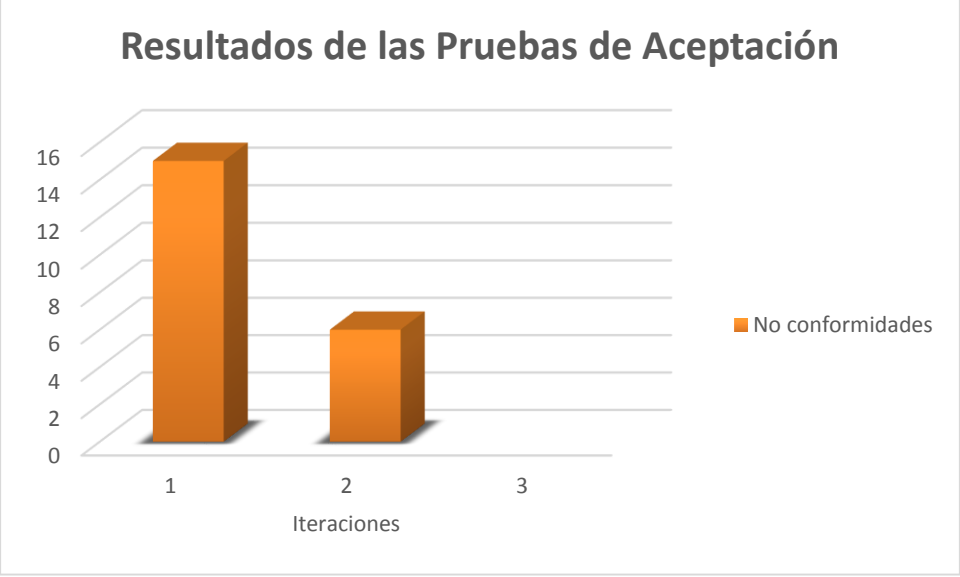

Ilustración 20: Resultados de las pruebas de aceptación.

3.7 Conclusiones del capítulo.

- En el capítulo se definieron los estándares de codificación que regirán la implementación de la propuesta de solución con el objetivo de lograr una mejor comprensión, análisis y mantenimiento del sistema, además de mayor organización.
- Se representan y describen los diagramas de componentes y despliegue con el fin de lograr un mejor entendimiento de la aplicación desarrollada además de aportar una vista lógica y real del proceso. Se describe cómo se comporta el tratamiento de errores en el sistema mediante los mecanismos que aporta el lenguaje de programación seleccionado.
- Se aplicaron pruebas de unidad y aceptación al sistema, registrándose el comportamiento del sistema, a partir de las distintas pruebas realizadas, se detectaron distintas no conformidades surgidas y corregidas en las iteraciones definidas, las cuales fueron de gran apoyo en el mejoramiento del sistema.

Conclusiones

En la presente investigación se analizaron las diversas problemáticas que afectan al proceso de solicitudes de trámites no presenciales de pasaportes en las entidades estatales y consulados. Se identificaron los métodos de investigación apropiados para realizar el estudio del campo de acción, tareas y objetivos.

- Se cumple con el objetivo propuesto al tener desarrollado un sistema informático que facilita la gestión de datos para las solicitudes de trámites no presenciales de pasaportes aportando seguridad al manejo y posterior traslado de las solicitudes, eliminando el uso de formato duro para el registro de las solicitudes, erradicando los volúmenes físicos de solicitudes y mejora los tiempos del proceso.
- El sistema desarrollado cumple con los requerimientos definidos convirtiéndola en una herramienta funcional que muestra resultados satisfactorios.
- Se documentaron los aspectos técnicos entorno al desarrollo del sistema, generando todos los artefactos y especificaciones recomendadas por la metodología de desarrollo XP.
- El presente trabajo constituye un significativo aporte al proceso de solicitudes de trámites de pasaportes realizados en las entidades estatales y consulados, ya que permite la informatización del proceso, contribuye con la seguridad de la información gestionada además de ahorrar recursos por concepto de gastos en materiales de oficina.

Por todo lo anteriormente expuesto se puede concluir que se dio cumplimiento al objetivo de la investigación y a los objetivos específicos, cumplimentándose el desarrollo del sistema de gestión de datos para las solicitudes de trámites no presenciales de pasaportes.

Recomendaciones

Concluida la investigación y basándose en las experiencias acumuladas durante el desarrollo del sistema, se proponen las siguientes recomendaciones con el objetivo de fortalecer la solución:

- Incorporar nuevas funcionalidades al sistema, con el objetivo de mejorar la validación de los datos biográficos registrados a partir de la integración del sistema con servicios del SUIN que garanticen la veracidad de la información.
- Establecer integración con dispositivos de captura para los datos biométricos.

Bibliografía referenciada

1. **OACI.** *Documentos de viaje de lectura mécanica (Doc 9303 Part1 Vol1).* 2006. Vol. 1.

2. **UNFPA.** Population & Development: UNFPA. Migration: A World on the Move. [En línea] 2010. [Citado el: 18 de Noviembre de 2013.] http://www.unfpa.org/pds/migration.html.

3. **Justicia, Ministerio de.** *GACETA OFICIAL DE LA REPÚBLICA DE CUBA.* La Habana : s.n., 2012. pág. 31. ISSN 0864-0793.

4. **Cento, Adrian Alberto Machado.** *Sistema Único de Identificación Nacional de la República de Cuba.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2013.

5. **Lamrani, Salim.** Cuba forma hoy en un año más médicos que el total que tenía en 1959. *Cubadebate.* [En línea] 4 de Agosto de 2012. [Citado el: 6 de Mayo de 2014.] http://www.cubadebate.cu/opinion/2012/08/04/cuba-forma-hoy-en-un-ano-mas-medicos-que-el-total-que-tenia-en-1959/.

6. **ESPAÑOLA, REAL ACADEMIA.** trámites. [En línea] [Citado el: 5 de Diciembre de 2013.] http://lema.rae.es/drae/?val=tr%C3%A1mites .

7. **Interior, Ministerio del.** *Requerimientos funcionales para la personalización centralizada a solicitudes de pasaportes.* 2013.

8. **ALEGSA.** Diccionario de informática. *ALEGSA.* [En línea] [Citado el: 1 de Abril de 2014.] http://www.alegsa.com.ar/Dic/encriptacion.php.

9. **López, Manuel José Lucena.** Cripotografía y Seguridad en Computadores. *Sunshine.* [En línea] 4ta Edición, Versión 0.7.0. [Citado el: 1 de Abril de 2014.] http://sunshine.prod.uci.cu/book/5137eec90571741de4000008/.

10. **Digitales, Departamento de Sistemas.** Criptografía. *Entorno Virtual de Aprendizaje.* [En línea] [Citado el: 29 de Abril de 2014.] http://eva.uci.cu/file.php/92/Tema_2_Criptografia/Materiales/Apoyo/Fundamentos_basicos_de_la_criptografia_v2.1.pdf.

11. **McCaffrey, James.** Keep Your Data Secure with the New Advanced Encryption Standard. *MSDN Magazine.* [En línea] [Citado el: 30 de Abril de 2014.] http://msdn.microsoft.com/en-us/magazine/cc164055.aspx.

12. **Ministerio del Poder Popular para Relaciones Interiores, Justicia y Paz.** Servicio Administrativo de Identificación, Migración y Extranjería. [En línea] República Bolivariana de Venezuela. [Citado el: 16 de Enero de 2014.] http://www.saime.gob.ve/mision-y-vision/.

13. **Idia Herrera Rivero, Annie Cubas González.** Capa de Procesos del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de Venezuela. *Serie Científica de la Universidad de las Ciencias Informáticas.* [En línea] [Citado el: 21 de Marzo de 2014.] http://publicaciones.uci.cu/index.php/SC/article/view/1289. ISSN: 2306-2495.

14. **Software, Laboratorio Nacional de Calidad del.** *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS y CICLOS DE VIDA.* s.l. : Instituto Nacional de Tecnologías de la Comunicación (INTECO), 2009.

15. **Letelier, Patricio.** *Proceso de desarrollo de software.* s.l. : Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia.

16. **Letelier, Patricio y Penadés, Mª Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia, 2006.

17. **Roger S. Pressman, Ph.D.** *Ingeniería del Software, Un enfoque práctico.* Sexta Edición. s.l. : McGraw-Hill Interamericana, 2005. ISBN 970-10-5473-3.

18. **Roger S Pressman, Ph.D.** *Ingeniería del Software, Un enfoque práctico.* Séptima edición. s.l. : McGRAW-HILL INTERAMERICANA EDITORES, S.A. DE C.V., 2010. ISBN: 978-607-15-0314-5.

19. **Riola, Jose Carlos Carvajal.** *METODOLOGÍAS ÁGILES: HERRAMIENTAS Y MODELO DE DESARROLLO PARA APLICACIONES JAVA EE COMO METODOLOGÍA EMPRESARIAL.* Barcelona : s.n., 2008.

20. **Guerrero, Rafael Martínez.** PostgreSQL-es. [En línea] 2010. [Citado el: 9 de Diciembre de 2013.] http://www.postgresql.org.es/sobre_postgresql.

21. **Corporation, Oracle.** MySQL. [En línea] Oracle Corporation. [Citado el: 27 de Marzo de 2014.] http://www.mysql.com/.

22. —. OpenJDK. *JDK Features.* [En línea] ORACLE Corporation, 28 de Julio de 2011. [Citado el: 19 de Enero de 2014.] http://openjdk.java.net/projects/jdk7/features/.

23. **Oracle.** Java.net. *The Source for Java Technology Collaboration.* [En línea] Oracle, Project Kenai and Cognisyn. [Citado el: 19 de Enero de 2014.] https://www.java.net/.

24. **Gavin King, Christian Bauer, Max Rydahl Andersen, Emmanuel Bernard, y Steve Ebersole.** *HIBERNATE - Persistencia relacional para Java idiomático.* [En línea] 15 de Septiembre de 2010. [Citado el: 10 de Diciembre de 2013.] http://docs.jboss.org/hibernate/core/3.5/reference/es-ES/html_single/.

25. **Community, Netbeans.** Netbeans. [En línea] Oracle Corporation, 2013. [Citado el: 18 de Mayo de 2014.] https://netbeans.org/community/releases/74/.

26. **International, Visual Paradigm.** Visual Paradigm. [En línea] Visual Paradigm International. [Citado el: 19 de Enero de 2014.] http://www.visual-paradigm.com/.

27. **Siegel, Jon.** Introduction to OMG's Unified Modeling Language™ (UML®). [En línea] 18 de Abril de 2013. [Citado el: 19 de Enero de 2014.] http://www.omg.org/gettingstarted/what_is_uml.htm.

28. **Sommerville, Ian.** *Ingeniería del Software.* Séptima Edición. Madrid : PEARSON EDUCACIÓN.S.A, 2005. pág. 712. ISBN:84-7829-074-5.

29. **Larman, Craig.** Entorno Virtual de Aprendizaje. *UML y Patrones,Introducción al análisis y diseño orientado a objetos.* [En línea] [Citado el: 4 de Marzo de 2014.] http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/UML_y_Patrones. ISBN 970-17-0261-1.

30. **Roger S. Pressman, Ph.D.** *Ingeniería del Software, Un enfoque práctico.* Quinta edición. s.l. : McGRAW-HILL INTERAMERICANA, 2002.

31. **Unidad Docente de Ingeniería del Software, Facultad de informática - Universidad Politécnica de Madrid.** Patrones del “Gang of Four”. [En línea] [Citado el: 2 de Abril de 2014.] http://is.ls.fi.upm.es/docencia/proyecto/docs/patrones_gof.pdf.

32. **Computadoras, DDC Técnicas de Programación de.** Excepciones. *Entorno Virtual de Aprendizaje.* [En línea] [Citado el: 4 de Abril de 2014.] http://eva.uci.cu/file.php/69/Bibliografia/Manuales_de_la_asignatura/Cap_4_Excepciones.pdf.

33. **Group, Object Management.** Unified Modeling Language UML. [En línea] [Citado el: 10 de Diciembre de 2013.] http://www.uml.org/.

34. **Labrín., Dr. Broderick Crawford.** Escuela de Ingeniería Informática. *PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO,Chile.* [En línea] 03 de Marzo de 2013. [Citado el: 21 de Mayo de 2014.] http://www.inf.ucv.cl/~bcrawford/AULA_ICI444/Pruebas.pdf.

Bibliografía consultada

1. **bsi.** ¿Qué son los sistemas de gestión? [En línea] [Citado el: 4 de Diciembre de 2013.] http://www.bsigroup.es/es/certificacion-y-auditoria/Sistemas-de-gestion/De-un-vistazo/que-son-sistemas-de-gestion/.

2. **Cuba.** EcuRed. *Arquitectura en capas.* [En línea] 2009. [Citado el: 7 de Diciembre de 2013.] http://www.ecured.cu/index.php/Arquitectura_en_Capas.

3. **Authority, SQL.** Journey to SQL Authority with Pinal Dave SQL, SQL Server, MySQL, Big Data and NoSQL. [En línea] 9 de Diciembre de 2007. [Citado el: 9 de Diciembre de 2013.] http://blog.sqlauthority.com/2007/12/09/sql-server-acid-atomicity-consistency-.

4. **Cuba.** Ecured. *Proceso Unificado de Desarrollo.* [En línea] 2009. [Citado el: 10 de Diciembre de 2013.] http://www.ecured.cu/index.php/Proceso_Unificado_de_Desarrollo.

5. **Fernando, Diego.** Proyecto XP Gestión de Contratistas. *ANÁLISIS Y DESARROLLO DEL SISTEMA DE INFORMACIÓN PARA LA GESTIÓN DE CONTRATISTAS DEL SENA, "CENTRO DE GESTIÓN ADMINISTRATIVA Y FORTALECIMIENTO EMPRESARIAL" DE TUNJA.* [En línea] [Citado el: 12 de Diciembre de 2013.] http://rupgestioncontratistas.wordpress.com/.

6. **SSL.com.** [En línea] [Citado el: 13 de Enero de 2014.] http://info.ssl.com/article.aspx?id=10241.

7. **España.** SEHACESABER.ORG. *La seguridad de los pasaportes.* [En línea] Fundacion Edelvives, Comité Olímpico Español. [Citado el: 14 de Enero de 2014.] http://www.sehacesaber.org/noticia?idContenido=32044&edad=4.

8. **Cuba.** Ecured. *Biometría.* [En línea] [Citado el: 14 de Enero de 2014.] http://www.ecured.cu/index.php/Biometr%C3%ADa.

9. **Domingo Morales L, Javier Ruiz-del-Solar.** SISTEMAS BIOMÉTRICOS: MATCHING DE HUELLAS DACTILARES MEDIANTE TRANSFORMADA DE HOUGH GENERALIZADA. [En línea] [Citado el: 14 de Enero de 2014.] http://es.scribd.com/doc/160964168/SISTEMAS-BIOMETRICOS.

10. **España.** Diferencias en la visión del color. *Colores metaméricos.* [En línea] Ministerio de Educación,Cultura y Deporte. [Citado el: 16 de Enero de 2014.] http://recursos.cnice.mec.es/fp/artes/ut.php?familia_id=5&ciclo_id=1&modulo_id=6&unidad_id=191&menu_id=2286&padre_id=0&submenu_id=3136&pagestoyen=10&ncab=2.2.2&contadort=9.

11. **Foundations, Java.** [En línea] [Citado el: 1 de Abril de 2014.] http://javafoundations.blogspot.com/2010/07/java-estandares-de-programacion.html.

12. **Cuba.** EcuRed. *Arquitectura en tres niveles.* [En línea] 2009. [Citado el: 7 de Diciembre de 2013.] http://www.ecured.cu/index.php/Arquitectura_de_tres_niveles.

13. **Society, IEEE Computer.** *SWEBOK,Guide to the Software Engineering Body of Knowledge.* [ed.] École de technologie supérieure Pierre Bourque y Université du Québec à Montréal Robert Dupuis. Los Alamitos : Group Managing Editor, 2004. ISBN 0-7695-2330-7.

14. **Holzner, Steven.** *JAVA2.* s.l. : CORIOLIS.

15. **Christian Bauer, Gavin King.** *Hibernate in action, A guide to the concepts and practice of object/relacional mapping.* s.l. : Manning Publications Co, 2005. ISBN 1932394-15-X.

16. **Cockburn, Alistair.** *Crystal Clear,A Human-Powered Methodology For Small Teams, including The Seven Properties of Effective Software Projects.* 2004.

17. **Frank Buschmann, Regine Meunier,Hans Rohnert,Peter Sornmerlad,Michael Stal.** *PATTERN-ORIENTED SOFTWARE ARCHITECTURE, A System of Patterns.* Great Britain : WILEY, 2001. ISBN 0 471 95889 7.

18. **Mª Paloma Díaz, Susana Montero,Ignacio Aedo.** *Ingeniería de la web y patrones de diseño.* s.l. : PEARSON Prentice Hall. ISBN 84-205-4609-7.

Anexos

Anexo 1. Descripción de las Historias de Usuario.

HU_1 Permitir autenticación.

Historia de Usuario	
Número: HU_1	**Usuario:** desarrollador
Nombre de historia de usuario: Permitir autenticación.	
Prioridad en negocio: alta	**Iteración asignada:** 1
Riesgo en desarrollo: baja	**Puntos estimados:** 3
Programador responsable: Ariel Leandro Doimeadios Aguilera	
Descripción: Permite la autenticación de los usuarios autorizados a ingresar al sistema. Se utilizan los parámetros usuario y contraseña.	
Observación: Si existen errores en los datos el sistema alerta mostrando un mensaje de texto notificando que los parámetros no son válidos.	

Tabla 8: HU_1 Permitir autenticación.

HU_2 Gestionar usuario.

Historia de Usuario	
Número: HU_2	**Usuario:** desarrollador
Nombre de historia de usuario: Gestionar usuario.	
Prioridad en negocio: media	**Iteración asignada:** 1
Riesgo en desarrollo: baja	**Puntos estimados:** 2
Programador responsable: Ariel Leandro Doimeadios Aguilera.	
Descripción: Permite la gestión de los usuarios que tienen acceso al sistema, incluye datos personales de los usuarios.	
Observación: Gestionar usuario contiene insertar, modificar, inhabilitar usuario y buscar usuario.	

Tabla 9: HU_2 Gestionar usuario.

HU_4 Gestionar paquete de envío.

Historia de Usuario	
Número: HU_4	**Usuario:** desarrollador
Nombre de historia de usuario: Gestionar paquete de envío.	
Prioridad en negocio: alta	**Iteración asignada:** 1
Riesgo en desarrollo: baja	**Puntos estimados:** 3

Programador responsable: Ariel Leandro Doimeadios Aguilera.
Descripción: Permite gestionar los paquetes que se envían con las solicitudes de trámites de pasaportes.
Observación: Gestionar paquete de envió contienen crear, cifrar y exportar paquete. Se notifica si se envía correctamente el paquete mediante la ejecución de un mensaje de texto.

Tabla 10: HU_4 Gestionar paquete de envío.

HU_5 Gestionar reportes estadísticos.

Historia de Usuario	
Número: HU_5	**Usuario:** desarrollador
Nombre de historia de usuario: Gestionar reportes estadísticos.	
Prioridad en negocio: alta	**Iteración asignada:** 1
Riesgo en desarrollo: baja	**Puntos estimados:** 3
Programador responsable: Yoelmy Hernández Barzaga.	
Descripción: Permite gestionar reportes estadísticos según criterios de búsqueda.	
Observación: Gestionar reportes estadísticos incluye obtener listados estadísticos por pasaportes de envió y por rango de fecha. Se notifica si existen errores en la gestión de los reportes estadísticos mediante la ejecución de un mensaje de texto.	

Tabla 11: HU_5 Gestionar reportes estadísticos.

HU_6 Recepcionar respuesta de envío.

Historia de Usuario	
Número: HU_6	**Usuario:** desarrollador
Nombre de historia de usuario: Recepcionar respuesta de envío.	
Prioridad en negocio: alta	**Iteración asignada:** 1
Riesgo en desarrollo: baja	**Puntos estimados:** 3
Programador responsable: Yoelmy Hernández Barzaga.	
Descripción: Permite recepcionar las respuestas a los envíos realizados.	
Observación: Recepcionar respuesta de envío contiene importar y mostrar respuesta. Se notifica si existen errores en la recepción de las respuestas mediante la ejecución de un mensaje de texto.	

Tabla 12: HU_6 Recepcionar respuesta de envío.

Anexo 2. Prototipos de interfaz de usuario.

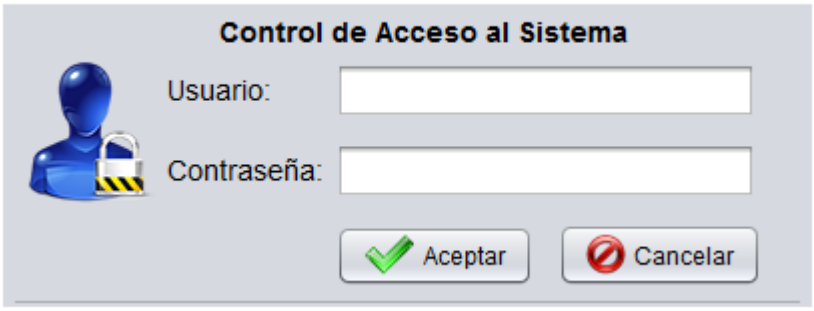

Ilustración 21: Prototipo de interfaz de usuario del RF1.

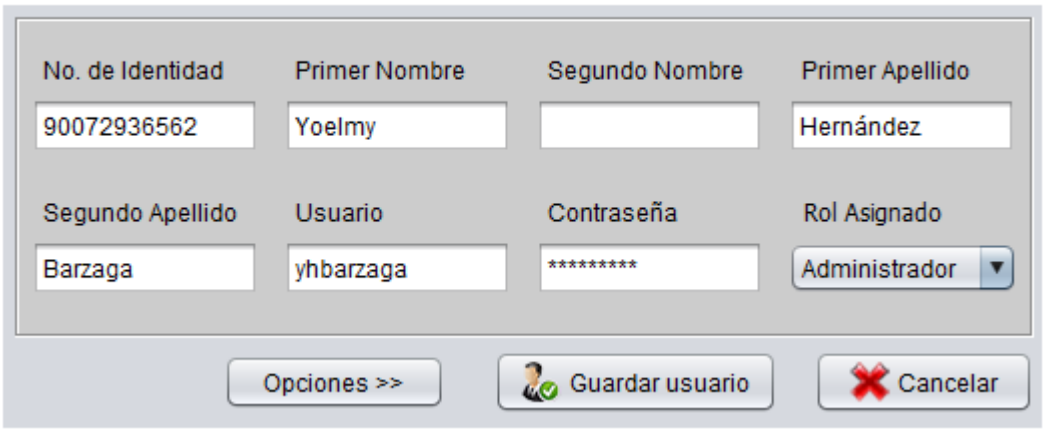

Ilustración 22: Prototipo de interfaz de usuario del RF2.1.

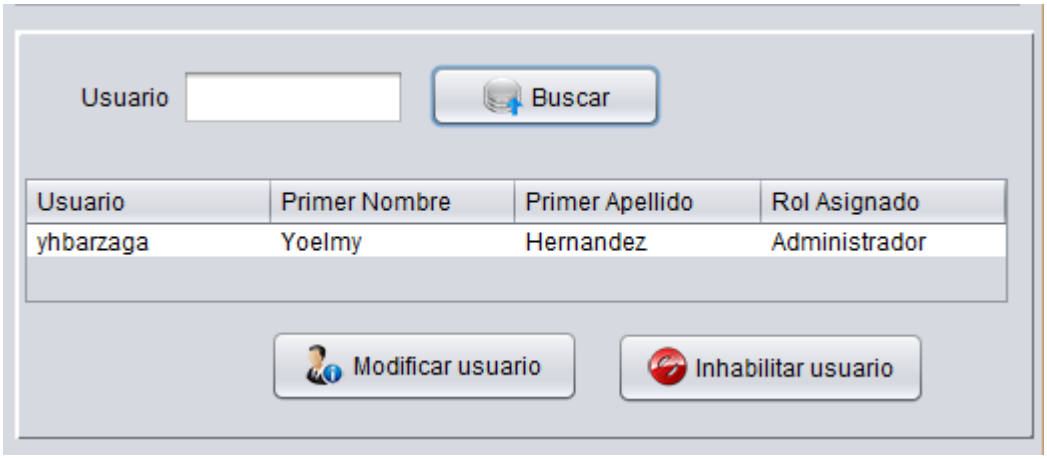

Ilustración 23: Prototipo de interfaz de usuario del RF2.2.

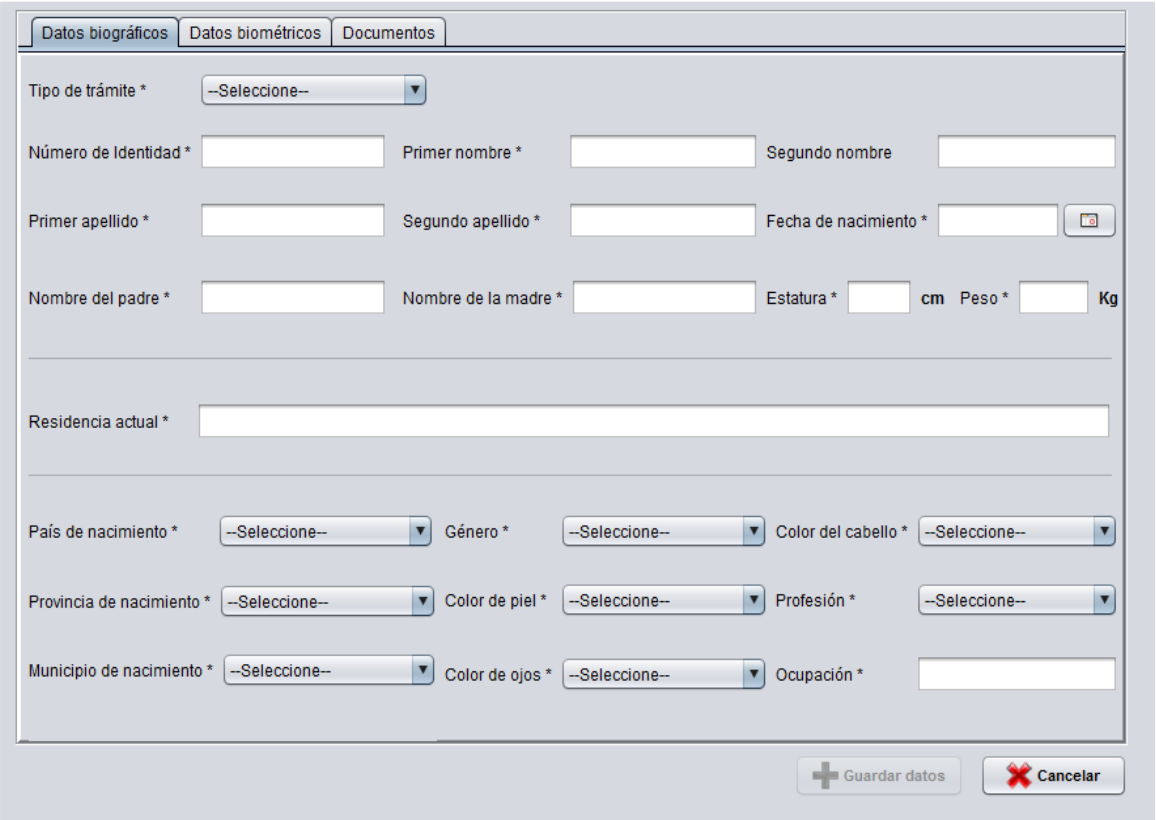

Ilustración 24: Prototipo de interfaz de usuario del RF 3.2.2.

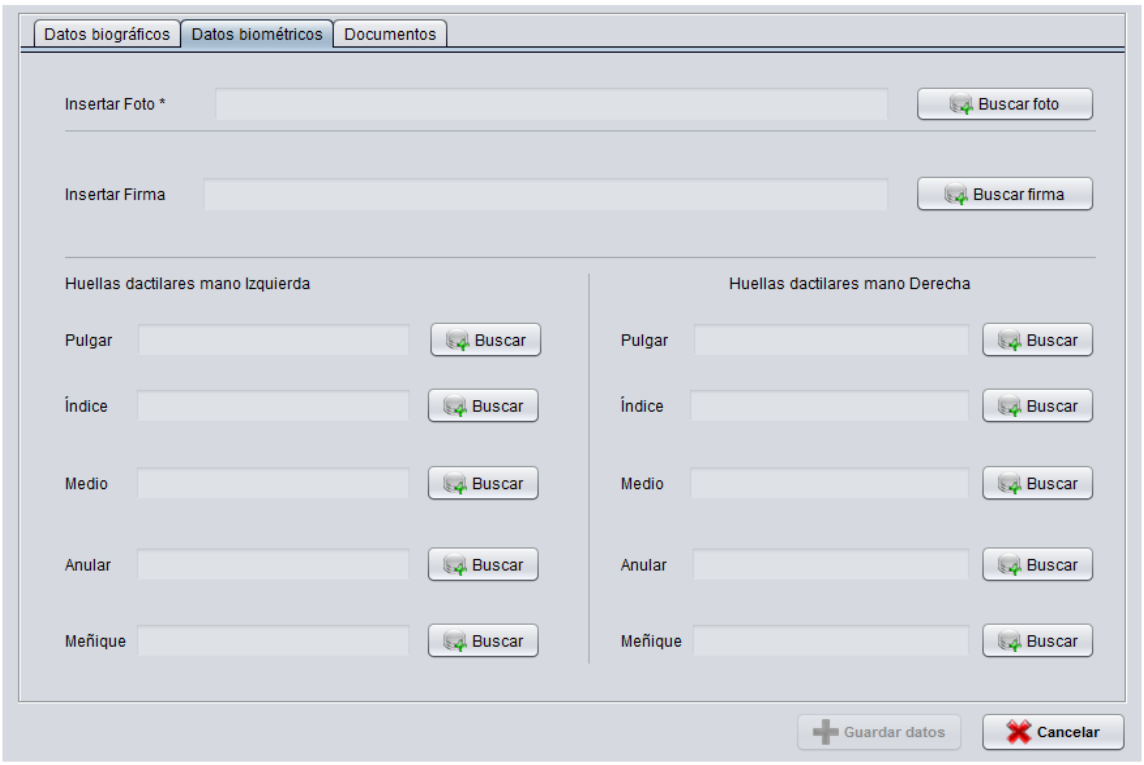

Ilustración 25: Prototipo de interfaz de usuario del RF 3.2.3.

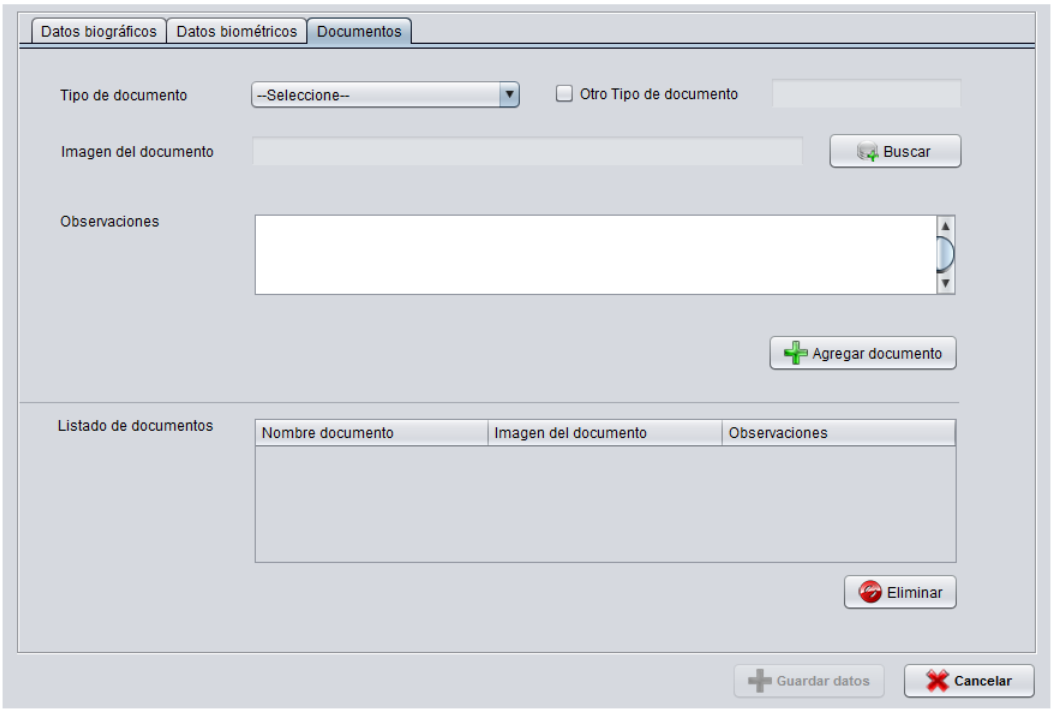

Ilustración 26: Prototipo de interfaz de usuario del RF 3.2.4.

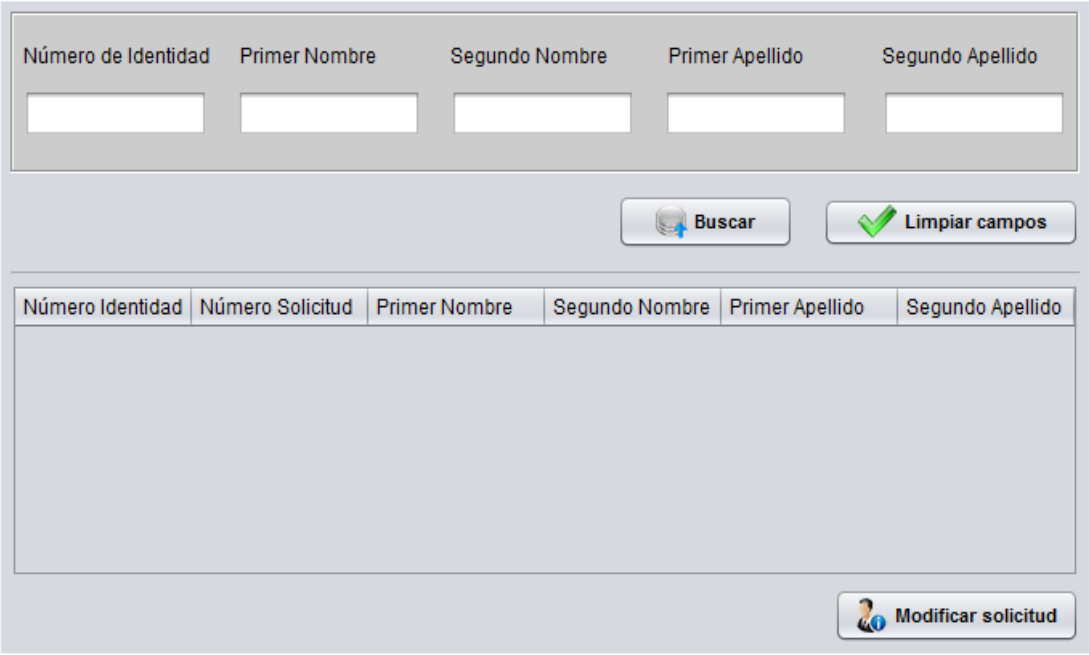

Ilustración 27: Prototipo de interfaz de usuario del RF3.3.

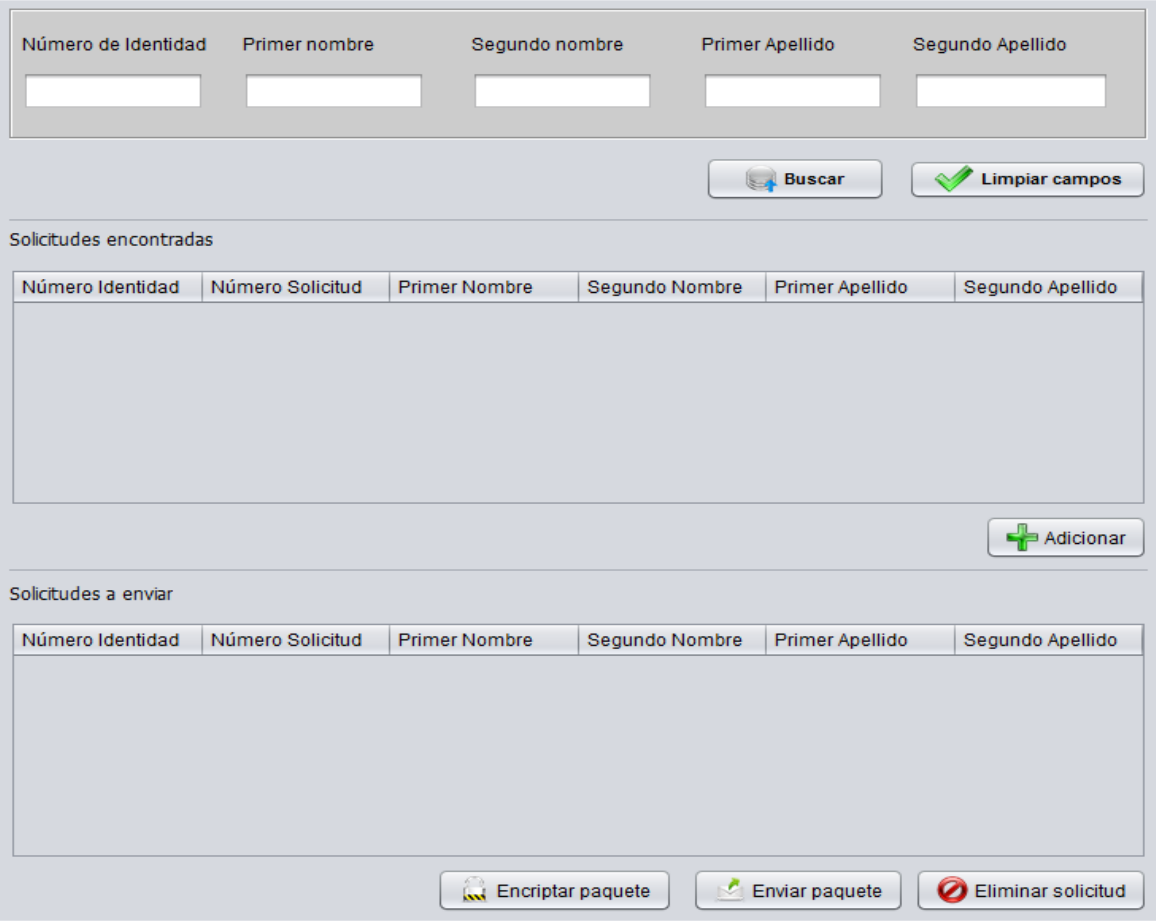

Ilustración 28: Prototipo de interfaz de usuario del RF 4.

Anexo 3. Tarjetas CRC.

HibernateFactory	
Responsabilidades	**Colaboradores**
Clase que trabaja en la capa de persistencia de datos, encargada de manejar las configuraciones de la conexión con la base de datos. Encargada de abrir y cerrar las sesiones de conexión cuando se necesite manipular datos.	import org.hibernate.HibernateException import org.hibernate.Session import org.hibernate.SessionFactory import org.hibernate.Transaction import org.hibernate.cfg.Configuration

Tabla 13: Tarjeta CRC de la clase HibernateFactory.

StnpEncriptarAES	
Responsabilidades	**Colaboradores**
Clase que trabaja en la capa donde se manejan las reglas del negocio. Encargada de cifrar y descifrar la información de las solicitudes, y de generar las claves correspondientes.	java.io.File java.io.FileInputStream java.io.FileNotFoundException java.io.FileOutputStream java.io.IOException java.io.InputStream java.io.OutputStream java.lang.System.in java.lang.System.out java.security.GeneralSecurityException java.security.InvalidKeyException java.security.Key java.security.KeyFactory java.security.KeyPair java.security.KeyPairGenerator java.security.NoSuchAlgorithmException java.security.PrivateKey java.security.PublicKey java.security.spec.InvalidKeySpecException java.security.spec.KeySpec java.security.spec.PKCS8EncodedKeySpec java.security.spec.X509EncodedKeySpec java.util.logging.Level java.util.logging.Logger javax.crypto.BadPaddingException javax.crypto.Cipher javax.crypto.CipherInputStream javax.crypto.CipherOutputStream javax.crypto.IllegalBlockSizeException javax.crypto.KeyGenerator javax.crypto.NoSuchPaddingException javax.crypto.SecretKey javax.crypto.spec.SecretKeySpec

Tabla 14: Tarjeta CRC de la clase StnpEncriptarAES.

StnpControlNegocioSolicitud	
Responsabilidades	**Colaboradores**
Clase encargada de atender todos los requerimientos del sistema. Clase controladora del sistema.	cu.uci.cised.stnp.acceso_datos.Ncolorcabello cu.uci.cised.stnp.acceso_datos.Ncolorpiel cu.uci.cised.stnp.acceso_datos.Imagenes cu.uci.cised.stnp.acceso_datos.Nmunicipio cu.uci.cised.stnp.acceso_datos.Ntipodocumento cu.uci.cised.stnp.acceso_datos.Ngenero cu.uci.cised.stnp.acceso_datos.Usuario cu.uci.cised.stnp.acceso_datos.Ntipotramite cu.uci.cised.stnp.acceso_datos.Solicitud cu.uci.cised.stnp.acceso_datos.Nprofesion cu.uci.cised.stnp.acceso_datos.Nprovincia cu.uci.cised.stnp.acceso_datos.Npais; cu.uci.cised.stnp.acceso_datos.Ncolorojos cu.uci.cised.stnp.acceso_datos.Persona cu.uci.cised.stnp.acceso_datos.Documento cu.uci.cised.stnp.acceso_datos.dao.DAO java.awt.Image java.io.ByteArrayInputStream java.io.ByteArrayOutputStream java.io.File java.io.FileInputStream java.io.IOException java.util.Date java.util.Iterator java.util.LinkedList java.util.List java.util.logging.Level java.util.logging.Logger javax.imageio java.beans.XMLDecoder java.io.BufferedInputStream java.io.BufferedReader java.io.FileNotFoundException java.io.FileReader java.io.UnsupportedEncodingException java.security.InvalidKeyException

	java.security.NoSuchAlgorithmException javax.crypto.BadPaddingException javax.crypto.IllegalBlockSizeException javax.crypto.NoSuchPaddingException javax.imageio.stream.ImageInputStream

Tabla 15: Tarjeta CRC de la clase StnpControlNegocioSolicitud.

IfrmPrincipal	
Responsabilidades	**Colaboradores**
Clase encargada del trabajo con la presentación al usuario. Encargada de permitir el acceso a las interfaces definidas para el usuario.	cu.uci.cised.stnp.negocio.StnpControlNegocioSolicitud cu.uci.cised.stnp.acceso_datos.Documento cu.uci.cised.stnp.acceso_datos.Imagenes cu.uci.cised.stnp.acceso_datos.Persona cu.uci.cised.stnp.acceso_datos.Solicitud cu.uci.cised.stnp.acceso_datos.Usuario java.util.LinkedList javax.swing.JOptionPane

Tabla 16: Tarjeta CRC de la clase IfrmPrincipal.

Anexo 4. Diagrama de clases.

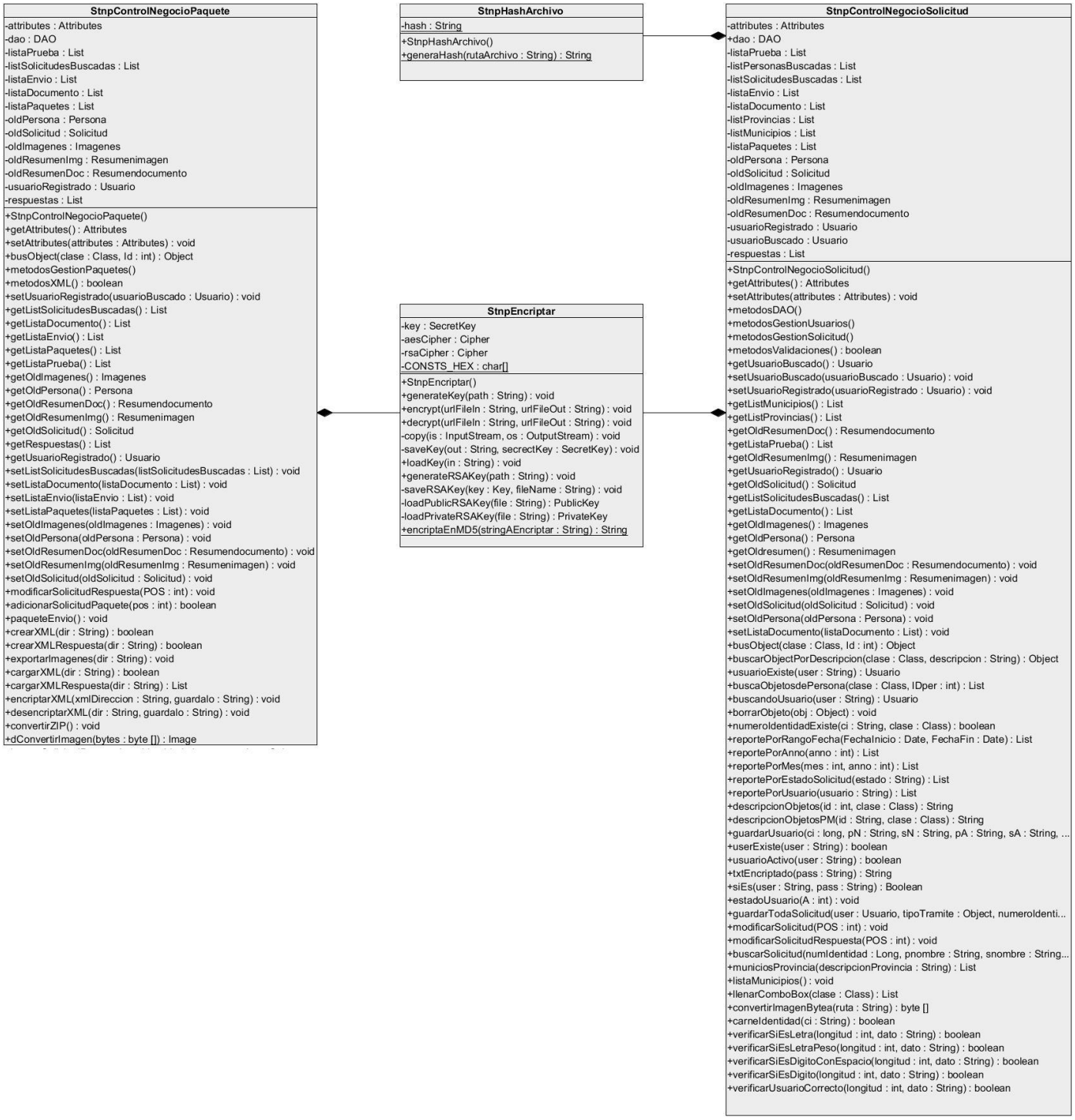

Ilustración 29: Diagrama de clases correspondiente al paquete negocio.

Anexo 5. Diagrama entidad-relación.

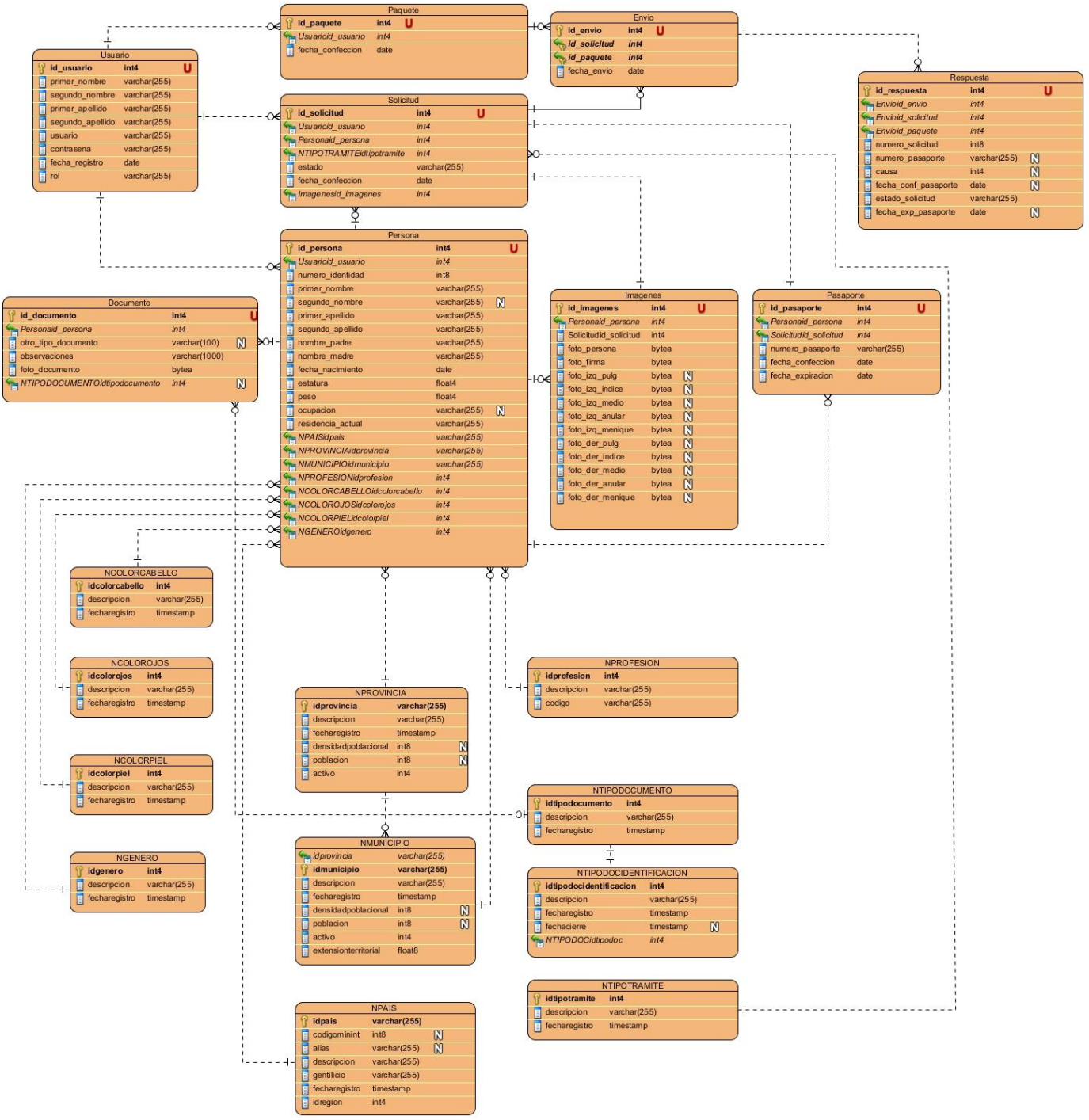

Ilustración 30: Diagrama entidad-relación.

Anexo 6. Plan de iteraciones.

Iteración	No. HU	Historias de Usuario	Duración estimada (semanas)
1	HU_1	Permitir autenticación	4
	HU_2	Gestionar usuario. • RF 2.1 Insertar usuario.	
	HU_3	Gestionar las solicitudes de pasaportes. • RF 3.1 Buscar persona. • RF 3.2 Insertar solicitudes de pasaporte. o RF 3.2.1 Definir tipo de trámite. o RF 3.2.2 Insertar datos biográficos de la persona. o RF 3.2.3 Insertar datos biométricos de la persona.	
	HU_4	Gestionar paquete de envío. • RF 4.1 Crear paquete.	
	HU_5	Gestionar reportes estadísticos. • RF 5.1 Obtener listados estadísticos por solicitudes enviadas.	
2			4
	HU_2	Gestionar usuario. • RF 2.2 Modificar usuario.	
	HU_3	Gestionar las solicitudes de pasaportes. • RF 3.3 Modificar solicitudes de pasaportes. o RF 3.3.1 Modificar datos biográficos de la persona.	
	HU_4	Gestionar paquete de envío. • RF 4.2 Cifrar paquete.	
3			3
	HU_2	Gestionar usuario. • RF 2.3 Inhabilitar usuario.	
	HU_3	Gestionar las solicitudes de pasaportes. • RF 3.3 Modificar solicitudes de pasaportes. o RF 3.3.3 Modificar documentos de la persona.	
	HU_4	Gestionar paquete de envío. • RF 4.3 Exportar paquete.	
	HU_5	Gestionar reportes estadísticos. • RF 5.2 Obtener listados estadísticos por rango de fecha.	
	HU_6	Recepcionar respuesta del envío. • RF 6.1 Importar respuesta.	

		• RF 6.2 Mostrar respuesta.	

Tabla 17: Plan de duración de las iteraciones.

Anexo 7. Plan de entregas.

Producto	Iteración 1	Iteración 2	Iteración 3
Sistema de gestión de datos para las solicitudes de trámites no presenciales de pasaportes.	20 de Marzo 2014	17 de Abril 2014	5 de Mayo 2014

Tabla 18: Plan de entrega.

Anexo 8. Tareas de ingenierías del sistema.

Iteración 2	
Historias de Usuario	**Tareas**

Gestionar usuario	• Modificar usuario.
	  ○ Buscar usuario a modificar.
	  ○ Validar campo de entrada de datos.
	  ○ Modificar datos del usuario.
	  ○ Ejecución de un mensaje de texto informando que se guardó satisfactoriamente el usuario.

Gestionar las solicitudes de pasaporte	• Modificar solicitudes de pasaporte. ○ Modificar datos biográficos de la persona. • Ejecución de un mensaje de texto informado que se guardó satisfactoriamente la solicitud.
Gestionar paquete de envío	• Cifrar paquete. • Ejecución de un mensaje de texto informado que se cifró satisfactoriamente el paquete.

Tabla 19: Tareas de ingeniería de la iteración 2.

Iteración 3	
Historias de Usuario	**Tareas**

Gestionar usuario	• Inhabilitar usuario.
	  ○ Buscar usuario.
	  ○ Inhabilitar usuario.
	  ○ Ejecución de un mensaje de texto informando que se inhabilitó el usuario.

Gestionar las solicitudes de pasaporte	• Modificar solicitudes de pasaporte.
	  ○ Modificar datos biométricos de la persona.
	  ○ Modificar documentos de la persona.
	• Ejecución de un mensaje de texto informado que se guardó satisfactoriamente la solicitud.

Gestionar paquete de envío	• Mostrar solicitudes según criterio de búsqueda. • Exportar paquete.

	• Ejecución de un mensaje de texto informado que se completó el proceso satisfactoriamente.
Gestionar reportes estadísticos	• Mostrar solicitudes según criterio de búsqueda: rango de fecha. ○ Se listan las solicitudes que coinciden con el criterio de búsqueda. • Ejecución de un mensaje de texto informado que la búsqueda produjo resultados o que no se encontraron solicitudes asociadas.
Recepcionar respuesta de envió	• Importar respuesta. • Mostrar datos asociados a la respuesta. • Ejecución de un mensaje de texto informado que se importó satisfactoriamente la respuesta.

Tabla 20: Tareas de ingeniería de la iteración 3.

Anexo 9. Pruebas unitarias del sistema.

```
/**
 * Test of txtEncriptado method, of class StnpControlNegocioSolicitud.
 */
@Test
public void testTxtEncriptado() {
    System.out.println("txtEncriptado");
    String pass = "yhbarzaga";
    StnpControlNegocioSolicitud instance = new StnpControlNegocioSolicitud();
    String expResult = "933bcc3240374339 59e063084dfbfc72";
    String result = instance.txtEncriptado(pass);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
}
```

Ilustración 31: Prueba unitaria realizada a la funcionalidad TxtEncriptado.

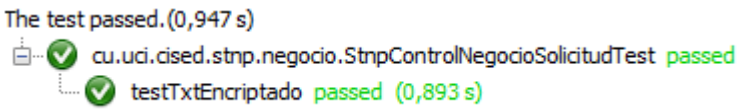

Ilustración 32: Resultado de la prueba unitaria realizada a la funcionalidad TxtEncriptado.

```java
/**
 * Test of descripcionObjetos method, of class StnpControlNegocioSolicitud.
 */
@Test
public void testDescripcionObjetos() throws NoSuchAlgorithmException, NoSuchPaddingException {
    System.out.println("descripcionObjetos");
    int id = 1;
    Class clase = Ncolorojos.class;
    StnpControlNegocioSolicitud instance = new StnpControlNegocioSolicitud();
    String expResult = "Pardos";
    String result = instance.descripcionObjetos(id, clase);
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
}
```

Ilustración 33: Prueba unitaria realizada a la funcionalidad descripcionObjetos.

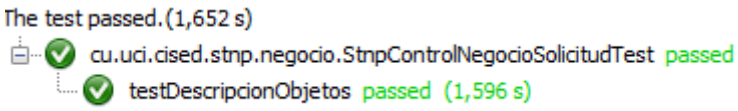

Ilustración 34: Resultado de la prueba unitaria realizada a la funcionalidad descripcionObjetos.

```java
/**
 * Test of find method, of class DAO.
 */
@Test
public void testFind() {
    System.out.println("find");
    Class clazz = Usuario.class;
    int id = 0;
    DAO instance = new DAO();
    Usuario expResult = new Usuario(2,"Yoelmy", "", "Hernandez", "Barzaga",
            "yhbarzaga", "933bcc3240374339 59e063084dfbfc72", new Date(), "Administrador",
            1,new Long("90072936562"));
    Usuario result = (Usuario) instance.find(clazz, id);
    assertEquals(expResult.getUsuario(), result.getUsuario());
    // TODO review the generated test code and remove the default call to fail.
}
```

Ilustración 35: Prueba unitaria realizada a la funcionalidad Find.

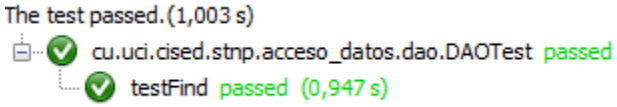

Ilustración 36: Resultado de la prueba unitaria realizada a la funcionalidad Find.

```
/**
 * Test of findUser method, of class DAO.
 */
@Test
public void testFindUser() {
    System.out.println("findUser");
    String user = "yhbarzaga";
    DAO instance = new DAO();
   Usuario expResult = new Usuario(2,"Yoelmy", "", "Hernandez", "Barzaga",
            "yhbarzaga", "933bcc324037433959e063084dfbfc72", new Date(), "Administrador",
            1,new Long("90072936562") );
   expResult.getUsuario();
    Usuario result = instance.findUser(user);
    assertEquals(expResult.getUsuario(), result.getUsuario());
    // TODO review the generated test code and remove the default call to fail.
}
```

Ilustración 37: Prueba unitaria realizada a la funcionalidad findUser().

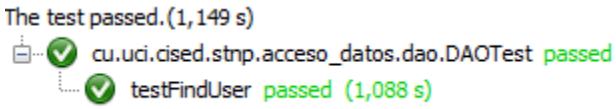

Ilustración 38: Resultado de la prueba unitaria realizada a la funcionalidad findUser().

Anexo 10: Casos de prueba de aceptación.

Casos de Prueba de Aceptación	
Código de casos de prueba: HU_2_CP2	**Nombre de la Historia de Usuario:** Gestionar usuario
Responsable de la prueba: Yoelmy Hernández Barzaga.	
Descripción de la prueba: Prueba de funcionalidad para la gestión de usuario. Dado los datos personales del usuario se deben poder insertar ese nuevo usuario, verificando que no existe previamente. Deber ser posible modificar los datos de un usuario existente así como poder inhabilitarlo.	
Condiciones de ejecución: La gestión de usuario solo está autorizado a realizarla el Administrador del sistema por lo que debe estar autenticado un usuario registrado con ese rol. Para registrar un nuevo usuario este no debe existir previamente. Para poder modificar los datos personales de un usuario este debe existir previamente.	
Entrada/Pasos de ejecución: 1. El usuario con rol de Administrador se autentica en el sistema. 2. Para registrar un nuevo usuario en el sistema:	

2.1. Se insertan los datos personales. 2.2. Se selecciona el rol. 2.3. Se guarda el nuevo usuario. 3. Para modificar un usuario del sistema: 3.1. Se busca el usuario deseado. 3.2. Se muestran los datos del mismo. 3.3. Se modifican. 3.4. Se guarda el usuario con los nuevos datos. 4. Para inhabilitar un usuario del sistema: 4.1. Se busca el usuario a inhabilitar. 4.2. Se inhabilita.
Resultado esperado: Para registrar un nuevo usuario: • Se registra el nuevo usuario, el sistema informa con la ejecución de un mensaje de texto. Para modificar un usuario del sistema: • Se modifican los datos del usuario seleccionado con los nuevos insertados, el sistema informa con la ejecución de un mensaje de texto. Para inhabilitar un usuario del sistema: • Se inhabilita el usuario seleccionado, el sistema informa con la ejecución de un mensaje de texto.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 21: Tabla A8_1. HU_2_CP2 Gestionar usuario.

Casos de Prueba de Aceptación	
Código de casos de prueba: HU_3_CP3_RF3.3	**Nombre de la Historia de Usuario:** Gestionar las solicitudes de pasaportes.
Responsable de la prueba: Yoelmy Hernández Barzaga.	
Descripción de la prueba: Prueba de funcionalidad para el requisito: Insertar solicitudes de pasaportes de la HU_3. Dado los datos biográficos, biométricos, y los documentos de la persona debe ser posible insertarlos.	
Condiciones de ejecución: La gestión de las solicitudes de pasaporte solo están autorizados a realizarla el Administrador del sistema y el que Redacta por lo que debe estar autenticado un usuario registrado con alguno de esos roles.	
Entrada/Pasos de ejecución: 1. El usuario con rol de Administrador o Redacta se autentican en el sistema. 2. Entrada de los datos biográficos: Número de Identidad, Primer Nombre, Segundo Nombre, Primer Apellido, Segundo Apellido, Nombre del Padre, Nombre de la Madre, País de nacimiento, Provincia/Municipio de nacimiento (si el país de nacimiento no es Cuba, se visualiza el campo Localidad), Género, Estatura, Color de ojos, Color de cabello, Color de la piel, Peso, Ocupación, Profesión y Residencia actual.	

3. Entrada de los datos biométricos: Fotos de la persona, Foto de la firma, Foto de las Huellas dactilares de ambas manos. 4. Entrada de los datos de los documentos: Tipo de documento (en caso de no existir se habilita el campo Otro Tipo de Documento), Imagen del documento y Observaciones. 5. Se guardan los datos biográficos, biométricos y documentos de la persona en la base de datos.
Resultado esperado: • Si los datos no son correctos cambia el color de fuente del campo que contiene la información incorrecta. • Si los datos son correctos el sistema informa mediante la ejecución de un mensaje de texto.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 22: HU_3_CP3_RF3.3 Gestionar las solicitudes de pasaportes.

Casos de Prueba de Aceptación	
Código de casos de prueba: HU_4_CP4	**Nombre de la Historia de Usuario:** Gestionar paquete de envío.
Responsable de la prueba: Ariel Leandro Doimeadios Aguilera.	
Descripción de la prueba: Prueba de funcionalidad para la gestión de los paquetes de envió. Se buscan las solicitudes que coincidan con el criterio de búsqueda seleccionado, se listan las solicitudes deseadas, se genera el fichero XML con la información de esas solicitudes, se encripta la información y se exporta.	
Condiciones de ejecución: La gestión de los paquetes de envió solo están autorizados a realizarla el Administrador del sistema y el que Redacta por lo que debe estar autenticado un usuario registrado con alguno de esos roles.	
Entrada/Pasos de ejecución: 1. El usuario con rol de Administrador o Redacta se autentican en el sistema. 2. Se ejecutan los criterios de búsqueda para obtener las solitudes deseadas. 3. Se listan las solicitudes seleccionadas para conformar el paquete. 4. Se genera el fichero XML con la información de esas solicitudes seleccionadas. 5. Se encripta la información contenida en el fichero. 6. Se exporta el paquete con el fichero(s) XML.	
Resultado esperado: • El sistema informa, mediante la ejecución de un mensaje de texto, que el proceso fue realizado satisfactoriamente o que ocurrió algún error.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 23: HU_4_CP4 Gestionar paquete de envío.

Casos de Prueba de Aceptación	
Código de casos de prueba: HU_5_CP5	**Nombre de la Historia de Usuario:** Gestionar reportes estadísticos.
Responsable de la prueba: Ariel Leandro Doimeadios Aguilera.	

Descripción de la prueba: Prueba de funcionalidad para la gestión de reportes estadísticos. Se buscan las solicitudes que coincidan con los criterios de búsqueda: Pasaportes por envío y Rango de fecha. Se muestran las solicitudes confeccionadas en un rango de fecha seleccionadas, tanto por año, mes, día o por sus combinaciones y por los envíos de pasaportes asociados a esas solicitudes.
Condiciones de ejecución: La gestión de los paquetes de envió solo están autorizados a realizarla el Administrador del sistema y el que Redacta por lo que debe estar autenticado un usuario registrado con alguno de esos roles.
Entrada/Pasos de ejecución: 1. El usuario con rol de Administrador o Redacta se autentican en el sistema. 2. Se selecciona el criterio de búsqueda. 3. Se muestran las solicitudes que coinciden con el criterio de búsqueda seleccionado.
Resultado esperado: • El sistema informa, mediante la ejecución de un mensaje de texto, que el proceso fue realizado satisfactoriamente o que ocurrió algún error.
Evaluación de la prueba:

Tabla 24: HU_5_CP5 Gestionar reportes estadísticos.

Anexo 11: Entrevista aplicada en el sector de la Salud Pública.

1) ¿Cantidad de solicitudes que tramitan por año?
2) ¿Cuántas etapas tiene el proceso de trámites de pasaporte y cuáles son?
   a) ¿Cómo funciona el proceso de trámite?
3) ¿Cantidad de personas que están implicadas en el proceso de captura y manejo de la información?
4) ¿Cómo almacenan las solicitudes y cómo las tramitan?
5) ¿Cuánto demora actualmente el proceso de trámite de las solicitudes?
6) ¿Costo que representa para el sector la confección de las solicitudes y su respectivo trámite?