Universidad de las Ciencias Informáticas

Facultad 3

Título: Componente Concertación del Sistema de Gestión para la Empresa de Seguros Nacionales.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: José Alberto Rios Fariñas

Tutores: Ing. Dinia Zayas Romero

Ing. Andy Daniel Meriño Coronado

Julio de 2016

DECLARACIÓN DE AUTORÍA

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

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

José Alberto Rios Fariñas

_____
Firma del Autor

Ing. Dinia Zayas Romero

_____
Firma del Tutor

Ing. Andy Daniel Meriño Coronado

_____
Firma del Tutor

DATOS DEL TUTOR

Contacto del tutor:

Ing. Dinia Zayas Romero

Síntesis del Tutor: Graduada como Título de Oro de Ingeniera en Ciencias Informáticas en el año 2007. Se ha desempeñado como Analista, Jefe de proyecto, Implantadora de soluciones y Jefa de Departamento en proyectos de gestión y actualmente como Analista del proyecto encargado del desarrollo de un sistema de gestión para la Empresa de Seguros Nacionales en el CEIGE.

Correo: dzayas@uci.cu
Teléfono: 7-835-8290

DATOS DEL TUTOR

Contacto del tutor:
Ing. Andy Daniel Meriño Coronado

Síntesis del Tutor: Graduado de Ingeniera en Ciencias Informática. Se desempeña como Arquitecto del proyecto ESEN en la actualidad.

Correo: admerino@uci.cu
Teléfono: 7- 837-3096

RESUMEN

Actualmente, en Cuba se realiza un gran esfuerzo en aras de mejorar los resultados económicos nacionales, para lo cual se hace uso de las Tecnologías de la Información y las Comunicaciones (TICs), en favor de lograr una mayor eficiencia y control sobres los recursos. La Empresa de Seguros Nacionales (ESEN) necesita un nuevo sistema que permita gestionar los procesos de comercialización de pólizas de seguro y dentro del mismo su proceso vital, la concertación de las pólizas de seguro.

Con este trabajo se propone desarrollar un componente que permita concertar las pólizas de seguro de la ESEN, tomando como guía para ello, la metodología de desarrollo utilizada en la Universidad de las Ciencias Informáticas (UCI), acorde a las peculiaridades de la entidad y las tendencias actuales del desarrollo de software. Con este resultado se obtiene un componente totalmente funcional como parte del Sistema Integral de Seguros Nacionales, que soporta el proceso de concertación de póliza que se lleva a cabo en la ESEN. Esta propuesta garantiza que se registre toda la información relacionada con la concertación de las pólizas de seguro de manera que esté accesible para el manejo de las mismas.

Palabras claves: gestión, concertación, póliza, seguros, sistemas

ÍNDICE

INTRODUCCIÓN

La actualización del Modelo Económico Cubano, puesto en marcha a partir de la aprobación de los Lineamientos de la Política Económica y Social de la Revolución, impone una utilización más racional del presupuesto. Es por ello que, dentro de las relaciones financieras, alcanza una dimensión especial el seguro, como mecanismo de transferencia de riesgos de los productores a las entidades aseguradoras mediante el pago de una prima de seguro. La prima constituye un pequeño porcentaje de los valores del patrimonio en riesgo con lo que se garantiza una recuperación ágil y efectiva, luego de eventos fortuitos que le provoquen daños o pérdidas. De esta manera se puede disminuir, en el nuevo modelo de gestión empresarial y en todo el sector productivo, el subsidio por pérdidas incluyendo las ocasionadas por desastres naturales.

Los elementos anteriores, sumados al incentivo que trae a la producción de renglones necesarios para el país, como el agropecuario, por ejemplo, por las características de un producto de seguro, hacen que este tema sea de interés social. Además, dentro de los objetivos de la actividad se encuentran la prevención de daños y la protección contra posibles pérdidas económicas, elementos todos muy importantes a tener en cuenta para el desarrollo del país.

En Cuba existen dos empresas aseguradoras, la Empresa de Seguros Nacionales (ESEN) y Seguros Internacionales de Cuba S.A. (ESICUBA). La gestión de seguros realizada por ambas se rige por las disposiciones del Decreto-Ley 263 del 2008 (Gaceta Oficial No. 005 / 2009), su Reglamento y el Decreto-Ley 177 sobre el ordenamiento del seguro y sus entidades.

La ESEN, como parte del Grupo Empresarial Caudal[1] perteneciente al Ministerio de Finanzas y Precios, es la empresa encargada de desarrollar operaciones de seguros y reaseguros[2] en moneda nacional y divisa tanto a personas naturales y jurídicas como extranjeras. Además, realiza actividades preparatorias y complementarias al seguro, dirigidas a la evaluación de riesgos y prevención de daños en moneda nacional y divisa a personas naturales y jurídicas cubanas y extranjeras. También le compete ofrecer servicios de inspección, tasación y ajustes de averías a través de contratos con terceros, los cuales son los encargados de ejecutarlos en la práctica. (ESEN, 2016)

Dicha entidad está dividida en Unidades Empresariales de Base (UEB) radicadas en cada una de las provincias del país, incluyendo el municipio especial Isla de La Juventud. En cada una de ellas se lleva a cabo un grupo de procesos sustantivos que constituyen el eje principal de la organización, dentro los cuales tiene un gran peso el proceso de concertación de pólizas teniendo en cuenta que

[1] **Grupo Empresarial Caudal:** Organización Superior de Dirección Empresarial (OSDE) de un conjunto de empresas dedicadas a la gestión y desarrollo de la actividad aseguradora, reaseguradora, financiera, de asistencia, inspección y ajuste de averías y de servicios conexos en general.

[2] **Reaseguro:** Contrato por el cual un asegurador toma a su cargo, total o parcialmente, un riesgo ya cubierto por otro asegurador, sin alterar lo convenido entre este y el asegurado.

tiene como objetivo la comercialización de pólizas de seguro de la ESEN de manera que se garantice la confianza y la satisfacción de los clientes.

En la actualidad, la ESEN tiene en explotación dos sistemas informáticos diferentes para la gestión de su información, los cuales están ubicados en las propias UEB del país, por lo que los informáticos tienen acceso a su código fuente y base de datos, permitiéndoles alterar su programación y la información almacenada, poniendo en riesgo su integridad. Además, existe falta de uniformidad en la información, lo cual trae como consecuencia la imposibilidad de conocer de forma ágil, por parte de la Oficina Central, datos necesarios como la cantidad de pólizas o asegurados a nivel de país, para la toma de decisiones, el seguimiento y el control de sus procesos.

En dichos sistemas, las bases de datos se encuentran descentralizadas y son distintas para cada uno de ellos, lo que posibilita que una persona se asegure en el mismo producto y con las mismas condiciones en dos provincias diferentes, lo cual constituye una violación a los procedimientos establecidos. Por otro lado, en ambas aplicaciones se cuenta como dos clientes distintos, a los asegurados que tienen dos contratos de seguro, cuando realmente es un asegurado con dos pólizas, lo cual obliga a realizar su gestión manual, pudiendo introducirse errores en los cálculos, lo que constituye un escenario favorable para las ilegalidades y provoca tardanza entre la confección de la documentación, y su aprobación a nivel central.

Teniendo en cuenta la importancia de los procesos que se llevan a cabo en la entidad y la necesidad de resolver las afectaciones existentes en la gestión de los mismos, surge, en noviembre de 2014, un proyecto de conjunto con la Universidad de las Ciencias Informáticas con el propósito de informatizar los procesos que se llevan a cabo en el área comercial de dicha empresa. Durante los siguientes meses de trabajo se identificaron 7 procesos, entre los cuales se encuentra la Concertación de las pólizas de seguro. A partir del trabajo realizado por los analistas del proyecto, se describieron los procesos y se identificaron un conjunto de requisitos funcionales y no funcionales para dar cumplimiento a las necesidades de informatización de la ESEN. Dicha labor produjo como resultado los artefactos necesarios para culminar la disciplina Requisitos, con la conformidad de los clientes expresada en un acta de aceptación de los mismos.

Teniendo como base lo expresado anteriormente se define como **problema** a resolver: las herramientas informáticas que se utilizan en la actualidad en la ESEN para la gestión del proceso de Concertación de pólizas de seguro presentan deficiencias que provocan la pérdida de información y dificultades en el acceso a la misma.

Para la solución de este problema se toma como **objeto de estudio:** el proceso de desarrollo de software para la gestión de pólizas de seguro, y como **campo de acción** el proceso de concertación de pólizas en los sistemas informáticos para la gestión de seguro.

Se define como **Objetivo General** de la investigación desarrollar el componente Concertación, cumpliendo con los requisitos identificados, de manera que se registre toda la información del proceso de Concertación de pólizas de seguros y permita el acceso a la misma.

Para lograr el objetivo propuesto, se han desglosado los siguientes objetivos específicos:

- Elaborar el marco teórico de la investigación que permita comprender el proceso de concertación de pólizas de seguro, así como el entorno donde se desarrollará la solución.
- Diseñar el componente Concertación basado en los requisitos previamente identificados.
- Validar el diseño del componente Concertación mediante el uso de métricas.
- Implementar el componente Concertación teniendo en cuenta el diseño realizado.
- Verificar y validar el componente obtenido mediante la realización de pruebas de software.

Con este trabajo se pretende obtener un componente que permita la concertación de pólizas a partir de los requisitos pactados con la ESEN y haciendo uso de las tecnologías, lenguajes y herramientas establecidas en el proyecto.

A continuación, se relacionan los métodos científicos utilizados en la investigación.

Métodos Teóricos

- ✓ Analítico – sintético: posibilita realizar el análisis de trabajos y documentos facilitando la obtención de conocimientos importantes para la investigación, permitiendo además obtener los elementos más significativos que se relacionan con el tema a desarrollar.
- ✓ Histórico – Lógico: se utiliza con el objetivo de investigar el funcionamiento y desarrollo del proceso de concertación de pólizas de seguro, así como de los sistemas de gestión del seguro.
- ✓ El método modelación: se utiliza para la descripción del proceso de concertación de pólizas de seguro.

El trabajo consta de 3 capítulos que a su vez se dividen en epígrafes y estos en subepígrafes de acuerdo con el nivel de detalle que requiere el contenido abordado en cada uno de ellos. A continuación, se explica brevemente el contenido de cada capítulo:

Capítulo 1: Fundamentación Teórica. En este capítulo se realiza un estudio sobre la gestión de pólizas de seguro en la ESEN, enmarcado en el proceso concertación. También se realiza un estudio de la metodología a utilizar, y con ello se definen las herramientas y lenguajes para generar los artefactos de sus disciplinas. Además, se realiza un estudio de los sistemas de gestión de pólizas existentes para potenciar el componente con sus ventajas.

Capítulo 2: Análisis y diseño del sistema. En este capítulo se describe el proceso de concertación de pólizas de seguro, se describen los requisitos funcionales y se generan los artefactos del diseño necesarios para la implementación del componente.

Capítulo 3: Implementación y pruebas. En este capítulo se describe los estándares utilizados para la implementación del componente, así como las especificaciones de su implementación y las garantías de su funcionamiento a través de las pruebas de software aplicadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se hace un estudio de la gestión del seguro en la ESEN, centrando la atención en la concertación de pólizas de seguro. También se describe la metodología de desarrollo, dando lugar a la definición de las fases, disciplinas, artefactos y herramientas a utilizar en el diseño e implementación del componente. Además, se realiza un estudio de los sistemas de gestión de pólizas existentes en aras de enriquecer el componente a desarrollar con sus ventajas.

1.2 La ESEN

Estructura

Actualmente, la ESEN, tiene en funcionamiento una Oficina Central, una Oficina de Negocios Especiales y una Oficina de Venta de Seguros, todas radicadas en La Habana; además, cuenta con 15 UEB a lo largo de todo el país. Dichas UEB, constituyen las representaciones del seguro en cada una de las provincias y controlan, a través de un cargo por municipio con categoría de director, a la red de agentes naturales y a las Cooperativas de Crédito y Servicios (CCS) que fungen como agentes jurídicos.

Ramos de seguro

Dicha entidad, se encarga de la comercialización de seguros en 3 ramos[3] fundamentales Seguros Personales, Seguros de Bienes y Seguros de Responsabilidad Civil, los cuales agrupan un conjunto de productos que se comercializan en todo el país.

- **Seguros Personales:** el objeto asegurado es la persona. Se protege al individuo ante la ocurrencia de un evento que le afecte directamente, como puede ser el fallecimiento, la supervivencia, la alteración de su salud o, en algunas ocasiones, su integridad psíquica. Este ramo contiene las siguientes agrupaciones de productos:

  - Seguro Temporario de vida: está diseñado para cubrir la necesidad económica que se produce ante el fallecimiento de una persona cuyos ingresos personales son determinantes en la economía familiar. Así como para aquellas personas que se incapacitan temporal o permanentemente, retirándola de sus funciones y actividades habituales con la correspondiente

[3] Ramo: conjunto de riesgos de características o naturaleza semejantes.

disminución o pérdida de los niveles de ingresos que tenían antes de incapacitarse.(ESEN - Temporario de Vida, 2016)

- ❖ Seguro de viajes: este seguro posee dos modalidades de contratación: Hacia el Exterior o Hacia Cuba. El mismo tiene como objetivo brindar protección financiera al Tomador del Seguro o al Asegurado, en caso de producirse alguno de los siguientes riesgos previstos y que originen la necesidad de incurrir en determinados gastos dentro del período de tiempo de duración de un viaje. (ESEN - Viajes, 2016)

  Riesgos:

  - ✓ Gastos médicos por enfermedad repentina o accidente.
  - ✓ Repatriación y/o transporte de heridos, enfermos y/o fallecidos.
  - ✓ Responsabilidad civil personal.
  - ✓ Anticipo de fianza judicial.
  - ✓ Asistencia en viaje, que comprende: Localización de Equipajes, Pérdidas Materiales, Pérdida de Documentos de Viaje y Transmisión de Mensajes Urgentes.

- **Seguro de bienes:** el objeto asegurado es el bien. Se protegen los bienes ante la ocurrencia de un evento que los afecte directamente. Este ramo contiene las siguientes agrupaciones de productos:

  - ❖ Seguro de Bienes Agrícolas: es un seguro destinado a proteger financieramente los cultivos y plantaciones permanentes bajo las modalidades de inversión y rendimiento. Los principales riesgos asegurables son: los daños y pérdidas ocasionadas por o a consecuencia de ciclón, tornado, aerovalanchas, manga de viento, inundación, lluvias intensas, granizo, sequía, incendio; así como plagas y enfermedades. (ESEN - Bienes agrícolas, 2016)
  - ❖ Seguro de Bienes Pecuarios: protege la muerte o sacrificio de los animales, así como la pérdida de su capacidad para cumplir el propósito a que estuvieron destinados al momento de suscribirse el contrato, siempre que los riesgos se produzcan por o a consecuencia de fenómenos climatológicos, enfermedades y accidentes. (ESEN - Bienes pecuarios, 2016)
  - ❖ Seguro de Bienes Patrimoniales Agropecuarios: protege las pérdidas o daños que sufren los bienes dedicados a la producción agropecuaria contra fuertes vientos asociados a ciclón, tornado, manga de viento, granizo, incendio por rayo y/o combustión espontánea, caída de objetos o aeronaves, deslizamientos del terreno, inundación imprevista e inevitable debido al desbordamiento del mar, ríos, lagunas y presas, caídas de sustancias tóxicas o químicas accidentalmente desde aviones. (ESEN - Seguros Patrimoniales Agropecuarios, 2016)

- ❖ Seguro de Vehículos de Transporte Terrestre: brinda cobertura a todas las personas naturales o jurídicas cubanas, extranjeras o sin ciudadanía, con residencia permanente en la República de Cuba, que son poseedoras legales de vehículos de motor. (*ESEN - Seguro de Vehículos de Transporte Terrestre* [no date])
- ❖ Seguro de incendio y líneas aliadas: está dirigido a proteger los riesgos que pueden afectar el inmueble y contenido que forma parte del patrimonio del Asegurado, por los daños o pérdidas causados directamente por Incendio, Rayo o Explosión, fenómenos meteorológicos y otras líneas aliadas. (ESEN - Seguro de Incendios, 2016)

- **Seguro de Responsabilidad Civil:** es aquel en el que el asegurador está obligado a indemnizar los daños y perjuicios causados a terceros, de los cuales sea civilmente responsable el asegurado, por un hecho y en los términos previsto en la póliza. Este ramo contiene la siguiente agrupación de productos:

  - ❖ Seguro de Responsabilidad Civil: este tipo de seguro protege al Asegurado por la responsabilidad civil, originada por daños a la propiedad ajena y lesiones o muerte. Existen seguros para los poseedores de licencia de conducción, licencia operativa para transporte de pasajeros y de carga, así como vehículos de tracción animal. (*ESEN - Seguro de Responsabilidad Civil*, 2016)

Procesos

La gestión de pólizas de seguro en la ESEN, está compuesta por 7 procesos generales, los cuales son concertación, seguimiento, reclamaciones, registro de ingreso, devolución de primas, control de instrumentos de cobro, y solicitud de servicio a terceros. Dichos procesos persiguen los siguientes objetivos:

- Concertación: comercializar las pólizas de seguro de la ESEN, de manera que se garantice la confianza y satisfacción de los clientes.
- Seguimiento: brindar seguimiento a las pólizas concertadas durante toda su vigencia.
- Reclamaciones: ejecutar las indemnizaciones ante la ocurrencia de un siniestro cubierto por la póliza, garantizando la satisfacción de los clientes.
- Registro de ingreso: registrar los ingresos monetarios obtenidos durante la gestión de las pólizas.
- Registro de primas: devolver el importe monetario recibido en exceso como prima.
- Control de instrumentos de cobro: supervisar los instrumentos de cobro de las UEB, conociendo en cada momento el estado en que se encuentran los mismos.

- Solicitud de servicio a tercero: realizar solicitudes a entidades de servicios auxiliares.

Teniendo en cuenta el campo de acción de la investigación, se centrará el estudio en el proceso de concertación, por lo que a continuación, se describen los principales conceptos vinculados al mismo.

- ❖ Persona jurídica: organización de personas o de personas y de bienes a la que el derecho reconoce capacidad unitaria para ser sujeto de derechos y obligaciones, como las corporaciones, asociaciones, sociedades y fundaciones.(DLE: persona, 2016)

- ❖ Asegurado: la persona titular del interés asegurado y, por consiguiente, aquella cuyos bienes, persona y responsabilidades están expuestas al riesgo y que ejerce los derechos y responde por las obligaciones de la relación contractual[4] constituida. (Gaceta Oficial No. 005 / 2009)

- ❖ Concertación: acuerdo entre dos o más personas o entidades sobre un asunto. Concierto, convenio. Acuerdo en común de las diversas partes que componen un todo. Arreglo o trato entre personas sobre un tema en particular. Ejemplo: Firman una nueva concertación petrolera. (Concertación, 2015)

- ❖ Póliza: documento en el que se hacen constar las condiciones del contrato de seguro y en la que se establecen los derechos y obligaciones de las personas que intervienen en él. (Gaceta Oficial No. 005 / 2009)

- ❖ Ramo: es el conjunto de riesgos de características o naturaleza semejantes. (Gaceta Oficial No. 005 / 2009)

- ❖ Tomador: la persona que no es el titular del interés asegurado, pero contrata el seguro, a nombre de un tercero, con la entidad de seguros. (Gaceta Oficial No. 005 / 2009)

- ❖ Beneficiario: aquel que es titular de los beneficios del contrato de seguro y que tiene acción directa contra la entidad de seguros, una vez acaecido el siniestro. (Gaceta Oficial No. 005 / 2009)

- ❖ Váucher: vale que da derecho a quien lo posee a adquirir determinados artículos o a disfrutar de un servicio. (DLE: váucher, 2015)

El estudio de los conceptos asociados al proceso, permitió conocer cómo, quién y que interviene en la concertación de pólizas de seguro, dando paso a la definición de la metodología de software para obtener la guía necesaria para el proceso de desarrollo del componente.

[4] **Contractual:** procedente del contrato o derivado de él.

1.3 Metodología de desarrollo de software

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, por lo que se utiliza el Modelo CMMI-DEV v1.3, ya que el mismo constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. (Metodología de desarrollo para la Actividad productiva en la UCI, 2015)
En este caso, se utiliza una variación de la metodología AUP, la cual rige los proyectos productivos en la UCI. Dicha metodología está compuesta por 3 fases principales: inicio, ejecución y cierre. (Metodología de desarrollo para la Actividad productiva en la UCI, 2015)

1.3.1 Fases de la metodología

I. Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
II. Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
III. Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Teniendo en cuenta que el proyecto tiene un ciclo de vida mayor a la investigación realizada, sólo se transitará por la fase de Ejecución.

1.3.2 Disciplinas de la metodología

Dicha metodología posee 7 disciplinas, al igual que AUP, pero a un nivel más atómico.

1. Modelado de negocio: El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes:

    i. Casos de Uso del Negocio (CUN).
    ii. Descripción de Proceso de Negocio (DPN).

iii. Modelo Conceptual (MC).

2. Requisitos: El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)].

   Teniendo como base que el modelado de negocio propone tres variantes a utilizar en los proyectos (CUN, DPN o MC) y existen tres formas de encapsular los requisitos (CUS, HU, DRP), surgen cuatro escenarios para modelar el sistema en los proyectos:

   - Escenario No 1: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que se puede modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización.

     Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.
     CUN + MC = CUS

   - Escenario No 2: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

     Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.
     MC = CUS

   - Escenario No 3: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.

     Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

DPN + MC = DRP

- Escenario No 4: aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

  Proyectos que no modelen negocio solo pueden modelar el sistema con HU.
  HU

3. Análisis y diseño: En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
4. Implementación: En la implementación, se construye el sistema, a partir de los resultados del Análisis y Diseño.
5. Pruebas internas: En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible, componentes de prueba ejecutables para automatizar las pruebas.
6. Pruebas de liberación: Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
7. Pruebas de Aceptación: Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Todas las disciplinas antes definidas (desde Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen Iteraciones y se obtengan resultados incrementales.

En una iteración se repite el flujo de trabajo de las disciplinas, Requisitos, Análisis y diseño, Implementación y Pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente.

1.3.3 Uso de las disciplinas de la metodología

A continuación, se describe el tránsito por las disciplinas, a partir de la captura de requisitos realizada por el proyecto.

En el modelado del negocio se obtuvo como resultado el funcionamiento del proceso concertación, y con ello las reglas de negocio del mismo, para lo cual se utilizaron los artefactos:

i. Descripción de Proceso de Negocio (DPN).
ii. Modelo Conceptual (MC).

1.3.3.1 Reglas del proceso de negocio concertación

Una regla del negocio se refiere a los hechos del sistema que se registran como datos y las restricciones sobre los cambios en los valores de esos hechos. Las perspectivas de las reglas del negocio implican el comportamiento de las personas en el sistema de negocios (actividad humana). Debido a que estas perspectivas son distintas, existe una definición de "reglas de negocio" para cada una de estas perspectivas. (BRG, 2016)

A continuación, se reflejan las reglas de negocio del proceso concertación de pólizas de seguro**,** las cuales se encuentran en el documento CEIGE-ESEN Reglas de Negocio.

- ✓ Por cada beneficiario se debe especificar el nombre, carné de identidad y el porciento del beneficio, el que se distribuirá a partes iguales si no se especifica algún valor.
- ✓ Las personas con edades entre 65 y 77 años solo pueden tener un máximo de 25000.00 pesos en la cobertura de muerte y de incapacidad permanente del producto Seguro Temporario de Vida
- ✓ En la contratación colectiva optativa, la suma asegurada por las coberturas de muerte e incapacidad permanente tiene un límite superior de 50,000.00 CUP. En pólizas individuales, las sumas aseguradas pueden estar por encima de 50000,00 pesos, siempre con la aprobación de la dirección de la UEB.
- ✓ Para los grupos familiares en las coberturas de muerte e incapacidad permanente se tendrá un límite mínimo de suma asegurada por persona de 5000.00 pesos para cada una de estas dos coberturas.

- ✓ La edad mínima establecida para contratar un seguro de viajes es de 18 años cumplidos, aunque los menores de 18 años pueden ser asegurados siempre que su póliza tenga un Contratante o Tomador.
- ✓ La edad mínima establecida para contratar un seguro de viajes es de 18 años cumplidos, aunque los menores de 18 años pueden ser asegurados siempre que su póliza tenga un Contratante o Tomador.
- ✓ El PTRC se crea en cada producto tomando como base las primas brutas mensuales ingresadas por monedas y modalidad de seguro.

1.3.3.1 Requisitos funcionales y no funcionales

En la disciplina requisitos se definió encapsular los requisitos utilizando Descripción de requisitos por proceso (DRP), y debido a esto se hizo uso del escenario 3, donde se obtuvo como resultado la agrupación de requisitos funcionales y los requisitos no funcionales del componente concertación.

Requisitos funcionales

Según (Sommerville, Galipienso 2005) los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.

A continuación, se ilustran las agrupaciones de requisitos funcionales obtenidas. Dichos requisitos funcionales se encuentran descritos en el documento *CEIGE ESEN Especificación de requisitos de software*.

La agrupación de requisitos se realizó siguiendo el criterio agrupación funcional, el cual agrupa los requisitos que tienen una relación funcional directa, por ejemplo, según el tipo de datos que se van a manejar o según los procesos de negocio que van a cubrir.

Agrupación de requisitos funcionales:

- ✓ Gestionar Asegurados.
- ✓ Gestionar beneficiarios.
- ✓ Gestionar agentes y comerciales.
- ✓ Gestionar tomador.
- ✓ Registrar póliza.

Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema, que incluyen restricciones de tiempo sobre el proceso de desarrollo y estándares. (Sommerville, Galipienso 2005)

Dichos requisitos se aplican al sistema en su totalidad, normalmente, apenas se aplican a características o servicios individuales del sistema. Los requisitos no funcionales forman una parte importante en el logro de la calidad de los productos de software, principalmente para que clientes y usuarios puedan valorar las características no funcionales del producto. Cumpliendo con toda la funcionalidad requerida entonces las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. (Sommerville, Galipienso 2005)

Las categorías de requisitos no funcionales, según Sommerville son:

Requerimientos del producto: Estos requerimientos especifican el comportamiento del producto. Algunos ejemplos son los requerimientos de rendimiento en la rapidez de ejecución del sistema y cuánta memoria se requiere; los requerimientos de fiabilidad que fijan la tasa de fallos para que el sistema sea aceptable; los requerimientos de portabilidad, y los requerimientos de usabilidad. (Sommerville, 2005)

Requerimientos organizacionales: Estos requerimientos se derivan de políticas y procedimientos existentes en la organización del cliente y en la del desarrollador. Algunos ejemplos son los estándares en los procesos que deben utilizarse; los requerimientos de implementación, como los lenguajes de programación o el método de diseño a utilizar, y los requerimientos de entrega que especifican cuándo se entregará el producto y su documentación. (Sommerville, 2005)

Requerimientos externos: Este gran apartado incluye todos los requerimientos que se derivan de los factores externos al sistema y de su proceso de desarrollo. Éstos pueden incluir los requerimientos de interoperabilidad que definen la manera en que el sistema interactúa con sistemas de otras organizaciones; los requerimientos legislativos que deben seguirse para asegurar que el sistema funcione dentro de la ley, y los requerimientos éticos. Estos últimos son puestos en un sistema para asegurar que será aceptado por sus usuarios y por el público en general. (Sommerville, 2005)

En la investigación se hace uso de la categoría requerimientos del producto propuesta por Sommerville, debido a que los requisitos están orientados hacia el comportamiento del componente.

Los requisitos no funcionales identificados son los siguientes:

Fiabilidad

- El componente solo permitirá la entrada de datos correctos.
- Se prevendrá la eliminación de información de la base de datos que se haya asociado a alguna operación.
- El componente mostrará en los mensajes de error los datos identificativos del lugar, operación o dato con error.

Usabilidad

- Se representarán los conceptos del dominio de la aplicación con un único ícono distintivo.
- El sistema permitirá acceder a todas las funcionalidades para entrar los datos y procesar la información a través de un menú.
- El componente permitirá la confirmación de la entrada de datos mediante el uso de mouse o teclado.
- El componente permitirá desplazarse por los campos de los formularios mediante el uso del tabulador.
- El componente responderá al 100% de los criterios seleccionados en las búsquedas.

En la disciplina análisis y diseño se refinaron los requisitos y el modelo conceptual para conseguir una comprensión más precisa y una descripción mejor estructurada.
Además, se definió utilizar la arquitectura cliente/servidor **Ver 1.3.2 Arquitectura de sistema**, y el patrón Modelo Vista Controlador **Ver 1.3.3 Patrón Modelo Vista Controlador (MVC).** siguiendo los estándares del proyecto, lo cual dio lugar a los lenguajes y herramientas a utilizar durante el diseño e implementación del componente.
Teniendo como base la forma de encapsulamiento de los requisitos, se definieron como artefactos a generar durante el diseño, los siguientes:

- Diagrama de proceso de negocio
- Modelo de datos
- Diagrama de clases del diseño

Una vez concluida la implementación del componente, se aplicarán los artefactos siguientes en la disciplina de pruebas:

- Diseño de casos de prueba: para las pruebas de caja negra.
- Técnica de partición equivalencia: para las pruebas de caja blanca.

Para la validación de la investigación se emitirá un acta de aceptación con la conformidad del cliente implícita.

1.3.4 Arquitectura de sistema

Arquitectura Cliente/Servidor.

La arquitectura Cliente/Servidor agrupa conjuntos de elementos que efectúan procesos distribuidos y computo cooperativo. Una aplicación típica es un servidor de base de datos al que varios usuarios realizan consultas simultáneamente. El proceso cliente realiza una consulta, el proceso servidor le envía las tablas resultantes de la consulta, el cliente las interpreta y muestra el resultado en pantalla. (ARQUITECTURA CLIENTE SERVIDOR, 2012)

Características de la Arquitectura Cliente/Servidor.

- El cliente y el servidor pueden actuar como una sola entidad o como entidades separadas, realizando actividades o tareas independientes.
- Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.
- Un servidor da servicio a múltiples clientes en forma concurrente.
- Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
- Tiene capacidad para integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.

Ventajas del uso de la Arquitectura Cliente/Servidor

El hecho de tener implementada este tipo de arquitectura posibilita que los clientes puedan hacer uso del sistema de gestión de pólizas, con el componente concertación implícito, en la misma máquina donde se encuentra el servidor o en distintas máquinas comunicadas a través de una red. Por lo

general, la parte de la aplicación correspondiente al cliente se optimiza para la interacción con el usuario, ejecutándose en su propia máquina, mientras que la parte correspondiente al servidor proporciona la funcionalidad multiusuario centralizada y se ejecuta en una máquina remota. Esta arquitectura hace que el mantenimiento del sistema sea fácil, debido a que las funciones y responsabilidades están distribuidas en diferentes equipos independientes, por lo que se puede reemplazar, reparar, actualizar, o trasladar un servidor, mientras sus clientes no son afectados por el cambio.

1.3.5 Patrón Modelo Vista Controlador (MVC)

Es un patrón arquitectónico con tres capas conceptuales. Este separa los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos, proporcionando diferentes vistas sobre un mismo modelo de datos y además facilita el mantenimiento en caso de errores. Tanto la vista como el controlador dependen del modelo, el cual no depende de otros conceptos o clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. Los tres elementos esenciales de este patrón son los siguientes:

- Modelo: es un conjunto de clases que se relacionan con el modelo del dominio[5], que tienen conocimiento de las vistas y que implementan los mecanismos necesarios para notificar a estas últimas sobre los cambios que se pudieren dar en el modelo del dominio. (Bascón Pantoja, 2004)
- Vista: son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo. (Bascón Pantoja, 2004)

Controlador: es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador. (Bascón Pantoja, 2004)

Entre las ventajas que tiene la utilización de este patrón arquitectónico se encuentran:

- Evita poner el código indebido en la capa impropia.

[5] Modelo del dominio: es el conjunto de clases que un ingeniero de software modela al analizar el problema que desea resolver.

- Soporta múltiples vistas, ya que la independencia que tiene el modelo con respecto a la vista permite que la interfaz de usuario pueda mostrar múltiples vistas de los mismos datos simultáneamente.
- Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Con la definición del patrón MVC las capas conceptuales quedaron conformadas de la siguiente manera:

Capa de presentación: ExtJS 6.0

ExtJS es un framework JavaScript que constituye una herramienta para la creación de aplicaciones de internet enriquecidas utilizando la tecnología AJAX. Tiene incluida una potente librería de componentes para formularios web que permite a los desarrolladores implementar funcionalidades de aplicaciones de escritorio como grids para mostrar datos, formularios, paneles, barras de herramientas, menús entre muchos otros permitiendo la reusabilidad de los mismos. Provee también una librería base para ejecutar funciones de JavaScript, manipulación de elementos del DOM, realizar peticiones AJAX y manejadores de eventos. Es compatible con múltiples plataformas de programación del lado del servido que pueden procesar peticiones POST tales como PHP, .Net y Java y puede ser ejecutado en la mayoría de los navegadores web utilizados actualmente como Internet Explorer versión 6 y posteriores, Mozilla Firefox, Opera y Safari. Este ofrece una gran cantidad de widgets (componentes de una interfaz de usuario) para crear interfaces de usuario complejas y funcionales. Posee un alto rendimiento, está bien diseñado y otorga un modelo de componentes extensibles. (Ext JS - JavaScript)

Capa de negocio: Spring Framework MVC 4.2.1

Dentro del framework de Spring se encuentra una parte referente al patrón Modelo Vista Controlador. Spring MVC se puede utilizar para crear aplicaciones web escalables y robustas. Este framework es altamente configurable vía interfaces y permite el uso de múltiples tecnologías para la capa vista como pueden ser JS1P y otros. Emplea la inversión de control para brindar una separación entre la lógica de los controladores y los objetos del negocio.

Capa de acceso a datos: Java

Es donde residen los datos y es la encargada de acceder a los mismos (WSO2). En el caso particular de esta solución está conformada por las clases que se encargan de realizar todo el almacenamiento de los datos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

1.3.6 Lenguajes de programación

Un lenguaje de programación está formado por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y significado de sus elementos y expresiones, es utilizado para controlar el comportamiento físico y lógico de una máquina. Es una técnica estándar que permite indicar las instrucciones que deben ser ejecutadas por una computadora, sobre qué datos deben operar y como estos deben ser almacenados y transmitidos. Estos se pueden dividir en lenguajes del lado del cliente y lenguajes del lado del servidor.

En la programación del lado del servidor se hará uso del framework Spring, como parte de la arquitectura del proyecto.

Spring Framework: Es un framework de código abierto. Se creó para abordar la complejidad del desarrollo de aplicaciones empresariales. Cualquier aplicación Java se puede beneficiar de Spring en términos de simplicidad, capacidad de prueba y acoplamiento. Proporciona una potente gestión de configuración basada en JavaBeans, además de una capa genérica de abstracción para la gestión de transacciones. (Aguirre Pérez, 2012)

Teniendo en cuenta la arquitectura definida, a continuación, se ilustran los lenguajes a utilizar:

Programación del lado del servidor

Java: es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems[6]. Hoy en día, existen muchas aplicaciones y sitios web que funcionan gracias a Java. Dicho lenguaje es rápido, seguro y fiable. Su utilización se extiende desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet. (Java)

Programación del lado del cliente

JavaScript: es un lenguaje de programación que se utiliza principalmente para crear páginas web capaces de interactuar con el usuario. Las páginas web se consideran estáticas cuando se limitan a mostrar un contenido establecido por su creador sin proporcionar más opciones al usuario que elegir

[6] **Sun Microsystems:** empresa informática que se dedicaba a vender estaciones de trabajo, servidores, componentes informáticos, software (sistemas operativos) y servicios informáticos.

entre los enlaces disponibles para seguir navegando. Cuando un creador incorpora JavaScript a su página, proporciona al usuario cierta capacidad de interacción con la página web, es decir, cierto dinamismo y por lo tanto se incrementan las prestaciones de la misma al añadir procesos en respuesta a las acciones del usuario. Estos procesos se ejecutan en la máquina del cliente (en el navegador) y por tanto no implican intercambio de datos con el servidor. (Mohedano, Saiz, Román 2012)

1.3.7 Herramientas para el diseño e implementación

A continuación, se enuncian las herramientas que conforman el entorno de integración continua implantado en el proyecto para el desarrollo del componente:

Visual Paradigm 8.0:

Es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. (Pressman Roger S, 2002)

Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software. (Pressman, 2002)

Entre sus principales características se encuentran:

- Soporta la última notación UML 2.5.
- Generación de bases de datos.
- Generador de informes para generación de documentación.
- Multiplataforma.
- Licencia: gratuita y comercial.

Herramientas de gestión y configuración de proyectos Maven:

Es una herramienta para la gestión y comprensión de proyectos Java. Es capaz de encargarse de la gestión de dependencias, construcción de los artefactos, generación de la documentación, etc. Su configuración es más simple y reutilizable. Una vez instalado y configurado, no es necesario mantenerlo. Es rápido el acceso a los paquetes ya descargados. Es independiente de una conexión a internet, excepto cuando se necesita algún paquete nuevo.

Hibernate 4.3.9:

Es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. (Rios, 2015)

Sencha Ext JS: es el marco más amplio MVC para construir aplicaciones web multiplataforma, ricas en funciones de orientación de escritorio, tabletas y teléfonos inteligentes. Ext JS aprovecha las funciones de HTML5 en los navegadores modernos, mientras que se mantiene la compatibilidad y funcionalidad para los navegadores antiguos. (Ext JS - JavaScript)

Gestor de base de datos: PostgreSQL 9.4.5

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.
(PostgreSQL-es, 2010)
PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (PostgreSQL-es, 2010)

Apache Tomcat 7.0.47:

Es un software desarrollado con Java (con lo cual puede funcionar en cualquier sistema operativo, con su máquina virtual java correspondiente) que sirve como servidor web con soporte de servlets y JSPs.

Entorno de desarrollo: Intellij IDEA 14.0:

Es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Cuenta con una enorme cantidad de extensiones y plugins, que pueden incorporarse al sistema para facilitar las tareas de programación. Además, tiene una intuitiva interfaz de usuario, un sistema de depuración de programas y las demás herramientas de ayuda, que facilitan el desarrollo de software. También cuenta dentro de sus funcionalidades con análisis de código, en busca de conexiones entre los símbolos a través de todos los archivos y los idiomas del proyecto, lo cual proporciona asistencia en profundidad de codificación, una navegación rápida, análisis de errores inteligente y refactorizaciones. (IntelliJ IDEA :: Features)

1.4 Sistemas para la gestión de seguros

A continuación, se realiza un estudio de los principales sistemas de gestión de pólizas de seguro existentes con el objetivo de estudiarlos e identificar sus ventajas en cuanto a facilidades de uso, accesibilidad a la información y servicios prestados para la gestión de pólizas de seguro.

Nekül:

Nekül es el primer sistema web que permite a los Productores Asesores de Seguro (PAS) realizar multicotizaciones y administrar de manera integral su cartera de clientes y pólizas, desde una única plataforma en línea gratuita. Además, los productores registrados en el sistema pueden acceder a Nekül desde cualquier computadora con conexión a Internet, siempre que así lo deseen. (NEKÜL, 2013)

Nekül ofrece las siguientes funcionalidades de manera totalmente gratuita y confiable:

- Multicotización a través de servicios web.
- Administración de cartera de clientes y pólizas.
- Control y seguimiento del estado de siniestros.
- Generación de reportes estadísticos y libros rubricados.

El sistema de multicotización de Nekül funciona de manera integrada a través de servicios web con 16 de las principales empresas aseguradoras a lo largo de Argentina. Dicha herramienta brinda la posibilidad a cada usuario de elegir hasta 3 compañías aseguradoras con las cuales cotizar en línea, manteniendo sus propias condiciones comerciales. A través de una interfaz de uso sencillo e intuitivo, Nekül permite a sus usuarios registrar y administrar pólizas de todos los ramos, en todos sus movimientos. Además, cuenta con diversos filtros de búsqueda que facilitan la localización de la información y generan un importante ahorro en los tiempos de trabajo. (NEKÜL, [2013)

Noa Gestión de Seguros:

NOA Gestión de seguros es un potente programa de gestión de seguros de todo tipo, especialmente pensado para Pymes que tienen un gran volumen de vehículos u otro tipo de pólizas y necesitan organizarlas y controlar los vencimientos, renovaciones de carné de conducir, ITV, visado de tarjetas, etc. Incluso el programa ofrece la posibilidad de añadir a cada ficha, una fotografía del vehículo y una copia de la ficha técnica, del carné de conducir.

También puede ser usado por agentes de seguros para controlar su cartera de pólizas, permitiendo llevar un control exhaustivo de todas las pólizas, vigilando fechas de vencimiento, cobros, renovaciones, historial de siniestros y cobertura. (NOA Gestión de seguros v5.0, 2016)

seQurnet:

El sistema seQurnet permite administrar de forma sencilla la cartera de un mediador de seguros. Comenzando por la facilísima importación directa de los archivos de pólizas y recibos que suministran las compañías. Continuando con el análisis comparativo de esta cartera con el ejercicio anterior. Haciendo el seguimiento del estado de esta cartera; cobros, devoluciones, anulaciones y finalizando con el proceso de las liquidaciones hacia las compañías y los colaboradores. Esta aplicación incorpora un potente archivo digital incrustado en los formularios de clientes, pólizas, recibos, siniestros, compañías y colaboradores, que permite almacenar documentos (pdf, jpg, doc, xls, msg, etc.) y que están accesibles de forma directa con un simple click. Además, incorpora plantillas personalizadas para cada usuario, por lo que el envío de comunicados a los clientes es muy cómodo. También permite enviar una carta personalizada directamente a los clientes y compañías, con un acabado totalmente profesional, mandar correos electrónicos vinculando los datos de sus pólizas y lanzar mensajes SMS[7] a través de la pasarela contratada. Dichos avisos podrán quedar registrados en el historial de cada cliente donde se programan notificaciones que se mostrarán en la agenda para su revisión posterior. Por otro lado, posee múltiples informes predefinidos y otros que cada usuario puede personalizar y obtener a la carta, como, por ejemplo; clientes que cumplan años un día concreto, clientes entre unas edades y con x años de permiso de conducir, clientes con 1, 2 o x pólizas, dando la posibilidad de elegir los datos a visualizar y de establecer los filtros de selección necesarios. (seQurnet v2.0, 2015)

GIS:

Programa informático para la gestión de corredurías y agencias de seguros. Cumple estrictamente las condiciones técnicas del nivel medio que se detallan en la Ley 15/1999 de Protección de Datos de Carácter Personal (LOPD)[8]. (GIS*, 2016*)

GIS dispone de los siguientes componentes:

- Gestión de seguros: permite gestionar propuestas, pólizas, recibos, siniestros, liquidaciones, listados, informes, etc. Además, dispone de un servicio de generación de informes para la Dirección General de Seguros (DGS). (GIS*,* 2016)
- Contabilidad: permite llevar la contabilidad para el usuario. Genera de manera automática los asientos contables asociados a las facturas y las cuentas contables asociadas a clientes y

[7] **SMS**: Mensaje corto de texto que se puede enviar entre teléfonos celulares o móviles.
[8] **LOPD:** Ley dictada en España por Juan Carlos I que tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar.

proveedores. Incluye además diario, cierre, balances, mayor, modelos de hacienda, listados, etc. (GIS, 2016)

INTERHELPER:

Es un programa informático diseñado para intermediarios de seguros, con el cual se puede mantener un completo control de los clientes y de las pólizas que estos han adquirido, los movimientos de dinero de todas las transacciones involucradas en una venta de seguros y sus renovaciones. Además, el programa de seguros InterHelper constituye una herramienta útil para realizar estrategias de mercadeo, atención a clientes, control de metas y resultados, administración de comisiones, y demás ventajas. Por otro lado, se adapta perfectamente a cualquier tipo de intermediario de seguros tales como corredores, agencias, franquicias, agentes independientes o colaboradores, sea cual sea su estructura y tamaño. Permite tener toda la información de los clientes, pólizas y vendedores en tiempo real, ingresar desde su casa u oficina, ver estadísticas y reportes de su negocio, entre otras. En dicho sistema los clientes ingresan mediante un formulario de acceso seguro insertado en la página web del intermediario y así pueden consultar información importante sobre sus propios productos contratados, imprimir copias de sus pólizas, recibos de pago, conocer vencimientos, cartera pendiente, siniestralidad, entre otras opciones. (InterHelper, 2016)

1.4.1 Valoración para la utilización de estos sistemas

Los sistemas de gestión de pólizas de seguro analizados poseen facilidades en cuanto a la administración de clientes y pólizas, brindando al cliente información detallada sobre las pólizas que haya concertado, lo cual proporcionó nuevas ideas en el desarrollo de las interfaces del componente. El indicador accesibilidad a la información denotó que varios de los sistemas son plataformas web accesibles a través de red, lo cual contrasta la necesidad de utilizar una arquitectura Cliente/Servidor. Mientras que el análisis de los servicios prestados, arrojó como resultado que no se corresponden con las funcionalidades a implementar, sin embargo, algunos de ellos proporcionaron ideas sobre implementación con respecto a la gestión de asegurados y el registro de pólizas de seguro.

1.5 Conclusiones del capítulo

En el presente capítulo se profundizó en el funcionamiento interno de la ESEN, conociendo así, la manera en que se lleva a cabo el proceso de concertación de pólizas en dicha entidad. También se conceptualizaron los términos asociados a la concertación de pólizas de seguro para comprender a cabalidad dicho proceso. Además, se estudió la metodología de desarrollo de software a utilizar, lo cual permitió definir las herramientas y lenguajes más adecuados, así como los artefactos necesarios para la investigación. Por otra parte, los sistemas de gestión de pólizas analizados ratificaron la

necesidad de implementar un nuevo componente de concertación de pólizas de seguro y proporcionaron nuevas ideas sobre diseño y codificación con la obtención de sus beneficios y debilidades.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

2.1 Introducción

En el presente capítulo se realiza el modelado del proceso de negocio concertación, dando lugar a la refinación del modelo conceptual del mismo. Se efectúa una propuesta de solución, a partir de la descripción de los requisitos funcionales, que detalla el componente para la concertación de pólizas de seguro de la ESEN. Además, se muestran los diferentes artefactos generados durante la fase, los cuales son evaluados a través de las métricas de validación del diseño.

2.2 Modelación del proceso de negocio

La modelación del proceso de negocio permite realizar una exploración del dominio del problema, con el fin de que el equipo de desarrollo comprenda el proceso que se realiza actualmente en la entidad.

El modelado del proceso de negocio es la base para comprender mejor la operación de una organización, documentar y publicar el proceso, buscando la estandarización y eficiencia de las operaciones dentro del proceso de la organización.

2.2.1 Proceso de negocio concertación

El proceso de concertación de pólizas de seguro se inicia a partir de la solicitud de un cliente, siendo el proceso responsabilidad del agente y trabajadores de las aseguradoras. Durante este se analizan los riesgos (realizado por el agente o una entidad tercera que brinde el servicio), se concerta el seguro, se cobra la prima, se entrega la póliza, se deposita el ingreso en el banco, se registra la nueva póliza y se elabora el expediente de seguro. Se crea también una provisión técnica que consiste en un importe monetario que respalda el riesgo en curso para el caso en que ocurriera, y se registra el ingreso y los movimientos contables asociados. Al concluir el proceso se tiene un documento oficial de la póliza que detalla las particularidades del contrato de seguro; el asegurado cuenta con los documentos que establecen sus deberes, derechos y constancias del pago realizado; la aseguradora cuenta con los documentos bancarios que amparan el depósito realizado en banco de los importes de prima, y el área contable de la empresa recibe un comprobante de operaciones. (Galafet Céspedes, 2015)

A continuación, se ilustra en la figura 1, el diagrama del proceso de negocio concertación.

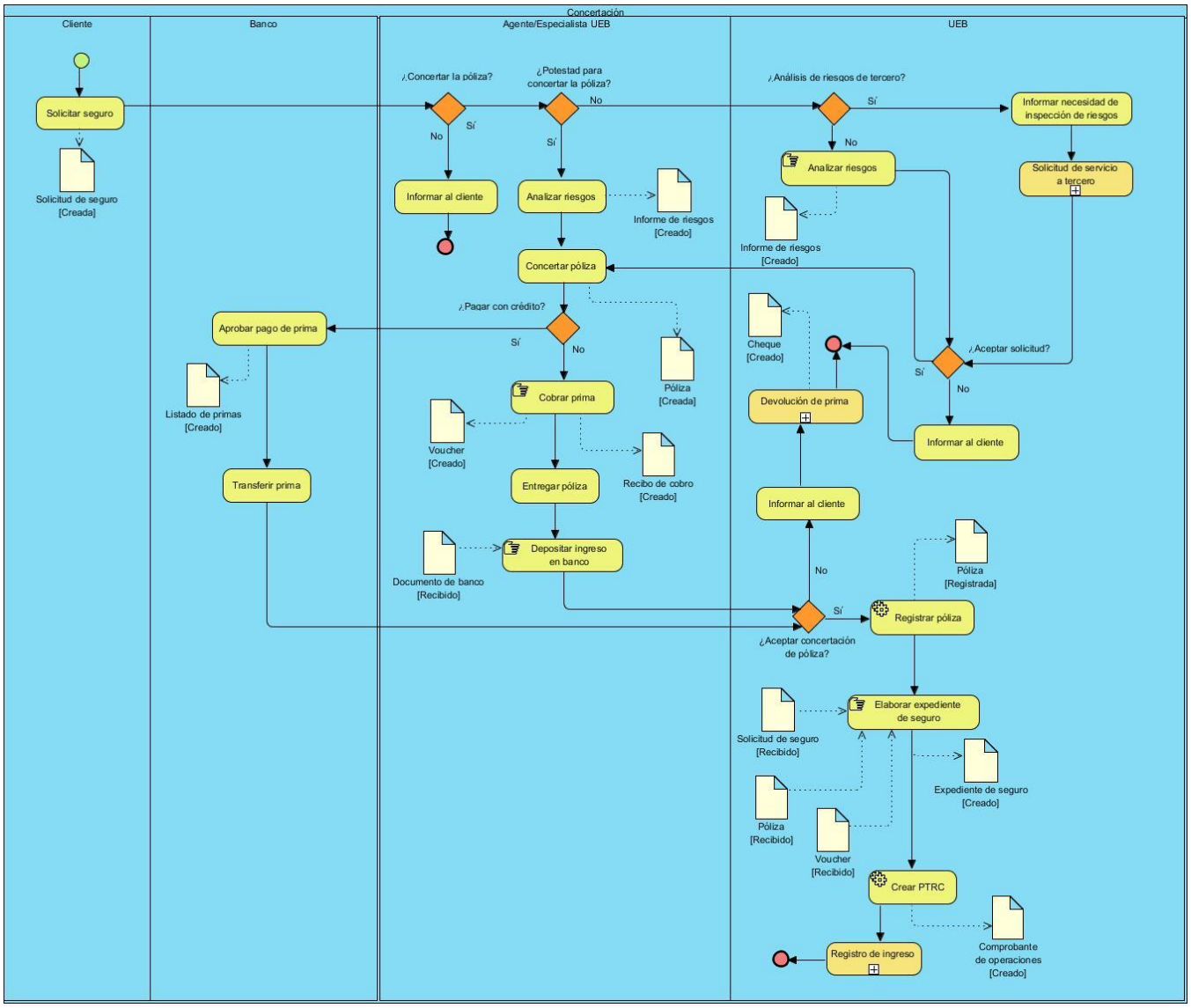


Figura 1: Diagrama del proceso de negocio concertación.

2.3 Modelo conceptual

El modelo conceptual se crea con el objetivo de proporcionar un marco analítico, estructurado y claramente definido de los principales conceptos del negocio y las relaciones existentes entre estos. Teniendo en cuenta que el negocio adquiere conceptos propios, se procedió a definir las relaciones existentes entre los términos identificados.

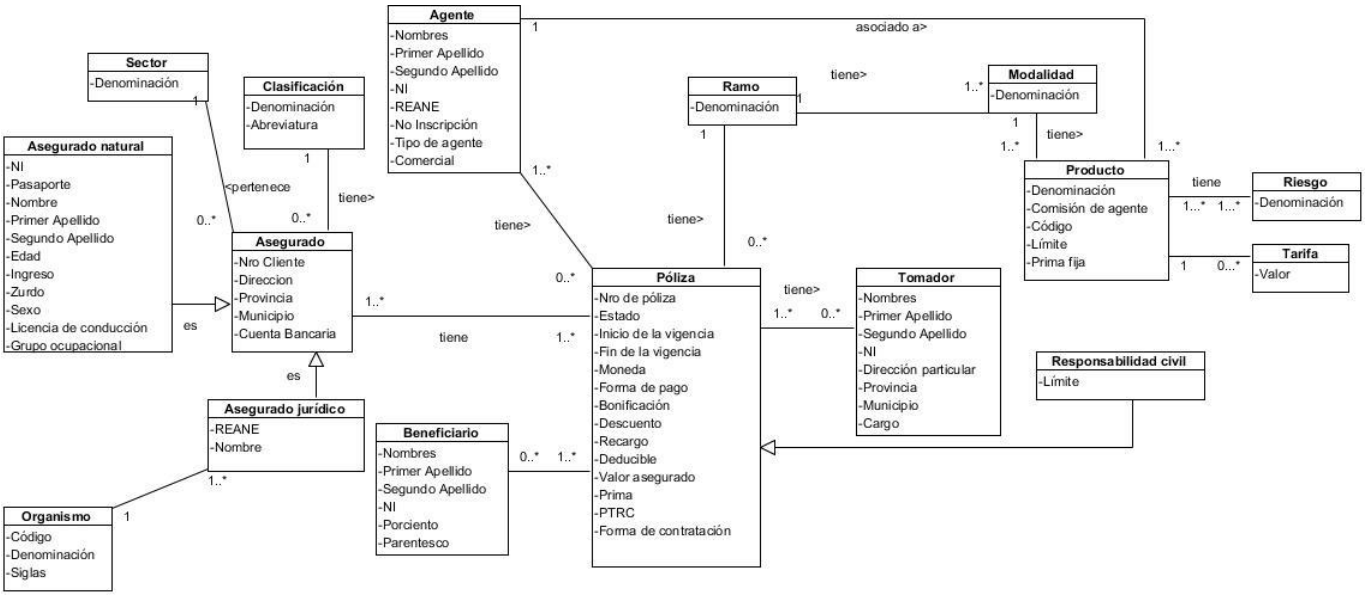


Figura 2: Modelo conceptual.

A continuación, se describen los conceptos definidos en el modelo conceptual:

Póliza: documento en el que se hacen constar las condiciones del contrato de seguro.
Persona natural: Persona con la capacidad de adquirir deberes y derechos jurídicos. Es una forma de representación legal dentro de la sociedad, independiente de la edad, el sexo o la religión.
Persona jurídica: persona ficticia, capaz de ejercer derechos y contraer obligaciones, y de ser representada judicial y extrajudicialmente.
Asegurado: persona cuyos bienes, persona y responsabilidades están expuestas al riesgo.
Beneficiario: persona titular de los beneficios del contrato de seguro.
Tomador: persona que contrata el seguro a nombre de un tercero.
Agente: toda persona natural o jurídica que expresamente autorizada y vinculada a una entidad de seguros sirve de mediador entre ésta y los tomadores.
Ramo: conjunto de riesgos de características o naturaleza semejantes.
Modalidad: subdivisión de los ramos para agrupar riesgos afines.
Riesgo: es una medida de la magnitud de los daños frente a una situación peligrosa.
Tarifa: monto de dinero que se paga por concepto del seguro contratado.
Responsabilidad civil: consecuencia de daños o perjuicios causados a terceros.

2.4 Descripción de los requisitos funcionales

En este epígrafe se realiza un desglose de las agrupaciones de requisitos funcionales mencionados en el capítulo anterior:

- La agrupación de requisitos **gestionar asegurados** incluye a los siguientes:
  - ✓ Adicionar asegurado: registra los asegurados en el sistema.
  - ✓ Modificar asegurado: modifica los datos de los asegurados registrados en el sistema.
  - ✓ Eliminar asegurado: elimina uno o varios asegurados registrados en el sistema.
  - ✓ Buscar asegurado: busca un asegurado en correspondencia con los criterios de búsqueda.
  - ✓ Listar asegurado: muestra una lista de los asegurados registrados en el sistema.
- La agrupación de requisitos **gestionar beneficiarios** incluye a los siguientes:
  - ✓ Adicionar beneficiario: registra los beneficiarios en el sistema.
  - ✓ Modificar beneficiario: modifica los datos de los beneficiarios registrados en el sistema.
  - ✓ Eliminar beneficiario: elimina uno o varios beneficiarios registrados en el sistema.
  - ✓ Buscar beneficiario: busca un beneficiario en correspondencia con los criterios de búsqueda.
  - ✓ Listar beneficiario: muestra una lista de los beneficiarios registrados en el sistema.
- La agrupación de requisitos **gestionar agentes y comerciales** incluye a los siguientes:
  - ✓ Adicionar agentes y comerciales: registra los agentes y comerciales en el sistema.
  - ✓ Modificar agentes y comerciales: modifica los datos de los agentes y comerciales registrados en el sistema.
  - ✓ Eliminar agentes y comerciales: elimina uno o varios agentes y comerciales registrados en el sistema.
  - ✓ Buscar agentes y comerciales: busca un agente o comercial en correspondencia con los criterios de búsqueda.
  - ✓ Listar agentes y comerciales: muestra una lista de los agentes y comerciales registrados en el sistema.

- La agrupación de requisitos **gestionar tomador** incluye a los siguientes:

  - ✓ Adicionar tomador: registra los tomadores en el sistema.
  - ✓ Buscar tomador: busca un tomador en correspondencia con los criterios de búsqueda.

- La agrupación de requisitos **registrar póliza** incluye a los siguientes:

  - ✓ Registrar póliza: registra las pólizas en el sistema.
  - ✓ Listar póliza: muestra una lista de las pólizas registradas en el sistema.
  - ✓ Asociar asegurado a la póliza: asocia uno o varios asegurados a la póliza que se está registrando en el sistema.
  - ✓ Asociar beneficiario al asegurado: asocia uno o varios beneficiarios a un asegurado.
  - ✓ Asociar tomador a la póliza: asocia un tomador a la póliza que se está registrando en el sistema.
  - ✓ Calcular prima: calcula la prima referente a la póliza que se está registrando en el sistema.

2.5 Arquitectura de diseño

Para dar cumplimiento a los requisitos funcionales es necesario obtener el diseño de la arquitectura del sistema y el diseño de la arquitectura de datos, debido a que a partir de estos procesos se definen términos claves para el desarrollo de la solución propuesta, haciendo uso de los artefactos diagrama de clases persistentes y diagrama de paquetes. Dichos diagramas permiten identificar dependencias entre varios componentes y las posibles integraciones a realizar, y también se obtiene el modelo de datos que permite definir el tamaño y el correcto diseño de la base de datos, siendo este uno de los puntos claves para el futuro desarrollo de la aplicación.

2.5.1 Modelo de datos

Una base de datos relacional es una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas desde la que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base. (SearchDataCenter, 2015).Tiene como principios:

- Se compone de varias tablas o relaciones.
- No pueden existir dos tablas con el mismo nombre.
- Cada tabla es a su vez un conjunto de registros, filas o tupla.
- Cada registro representa un objeto del mundo real.
- Cada uno de estos registros consta de varias columnas, campos o atributos.
- No pueden existir dos columnas con el mismo nombre en una misma tabla.
- Los valores almacenados en una columna deben ser del mismo tipo de dato.
- Todas las filas de una misma tabla poseen el mismo número de columnas.
- La información puede ser recuperada o almacenada por medio de sentencias llamadas consultas.

2.5.1.1 Nomenclatura utilizada para la base de datos

Nombre de la base de datos:

El nombre de la Base de Datos (BD) comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing.
Ejemplo*:* Concertación

Nombre de las tablas:

El nombre de las tablas, debe escribirse con todas las letras en minúscula y con solo leerlo se reconoce el propósito de la misma. En caso de que el nombre contenga varias palabras estas se escribirán juntas. Además, el nombre de las tablas se escribe en singular y utilizando como prefijo las 4 primeras letras referentes al esquema al que pertenece.
Ejemplo: Esquema: concertación Tabla: conc_asegurado.

Nombre de las llaves primarias:

El nombre de las restricciones se escribe con minúscula. Para el caso de las claves primarias se utiliza el prefijo pk seguido del símbolo "_", y luego el nombre de la tabla sin especificar el prefijo de la misma.
Ejemplo: pk_poliza.

Nombre de las llaves foráneas:

El nombre de las llaves foráneas se escribe con minúscula, comenzará con el prefijo fk seguido del símbolo "_", luego el nombre de la tabla de donde viene, seguido del símbolo "_" y por último el nombre de la tabla que la recibe, con solo leerlo se entiende la idea. Ejemplo: fk_poliza_asegurado.

2.5.1.2 Diagrama de modelo de datos

A continuación, se presenta en la figura 2, el modelo de datos obtenido para la confección de la base de datos del componente, la cual se encuentra en tercera forma normal, debido a que no existen dependencias transitivas de atributos no llaves respecto a la llave primaria, y está en segunda forma normal. (García, 2005)

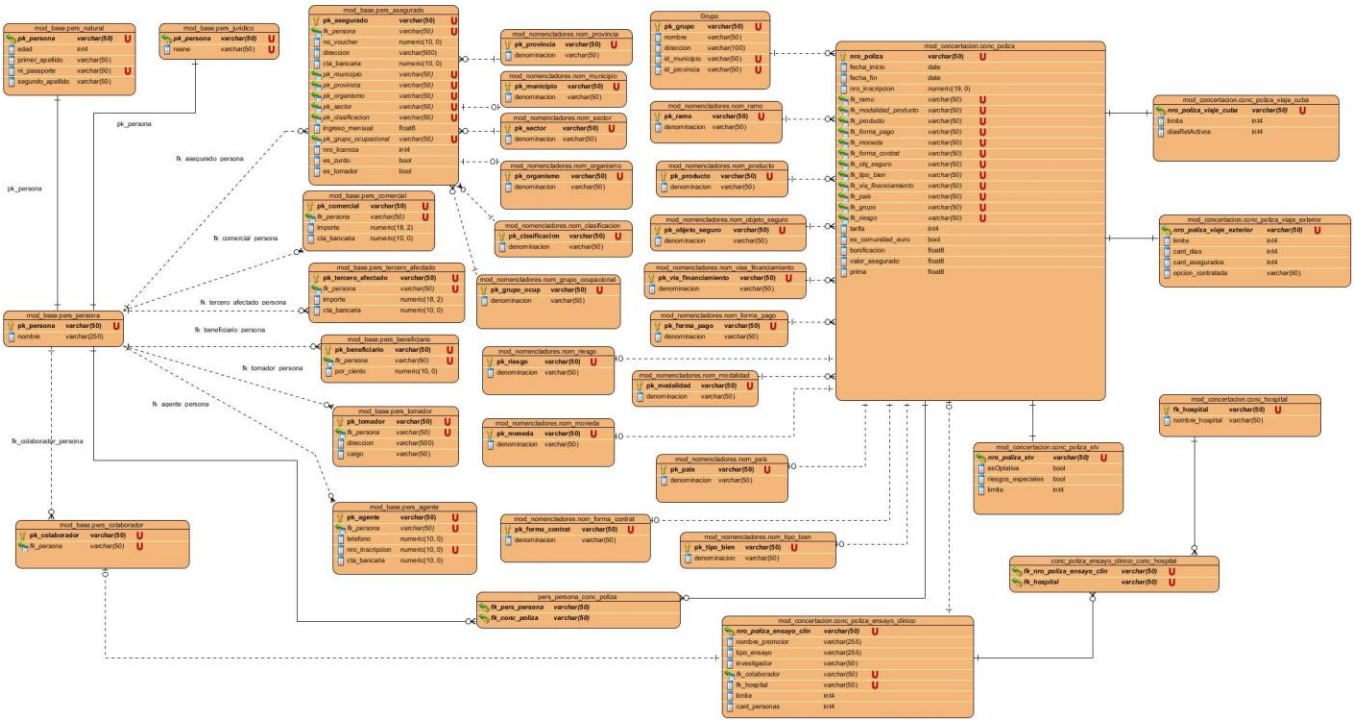


Figura 3: Modelo de datos.

2.5.1.3 Patrones de base de datos

El patrón **Representación de Objetos como Tablas** propone definir una tabla para cada clase de objeto persistente, donde los atributos de objetos que contienen tipos de datos (número, cadena, booleano y otros) se mapean en las columnas. (Larman, 2002)

Este patrón se utilizó en todas las tablas de la base de datos, debido a que se definió una tabla para cada clase persistente presente en el dominio del problema.

Ejemplo:

Tabla: mod_base.pers_persona

Clase persistente: Persona

El patrón **Identificador de Objetos (IDO)** propone asignar un ID0 a cada registro y objeto. Un identificador de objetos suele ser un valor alfanumérico, el cual es único para un objeto específico. Toda tabla de base de datos relacional tiene un ID0 como clave primaria, y los objetos también contarán (directa o indirectamente) con un identificador. (Larman, 2002)

Este patrón se utilizó en todas las tablas de la base de datos, debido a que se definió un identificador para cada una de ellas.

Ejemplo:

Tabla: mod_base.pers_persona

Identificador: pk_persona

2.5.2 Modelo de clases persistentes

Las clases persistentes representan almacenamientos de datos que persistirán más allá de la ejecución del software. En el diagrama se muestra la forma en que el componente obtiene datos de los flujos de trabajo de la base de datos a través de las clases de acceso, de manera que posibilita realizar tareas automáticas y semiautomáticas, estableciendo un estrecho vínculo con el marco de trabajo y guardando los resultados de las ejecuciones y los estados de las tareas en una base de datos. A partir del diseño del modelo de datos se elaboró el diagrama de clases persistentes que se presenta en la figura 3:

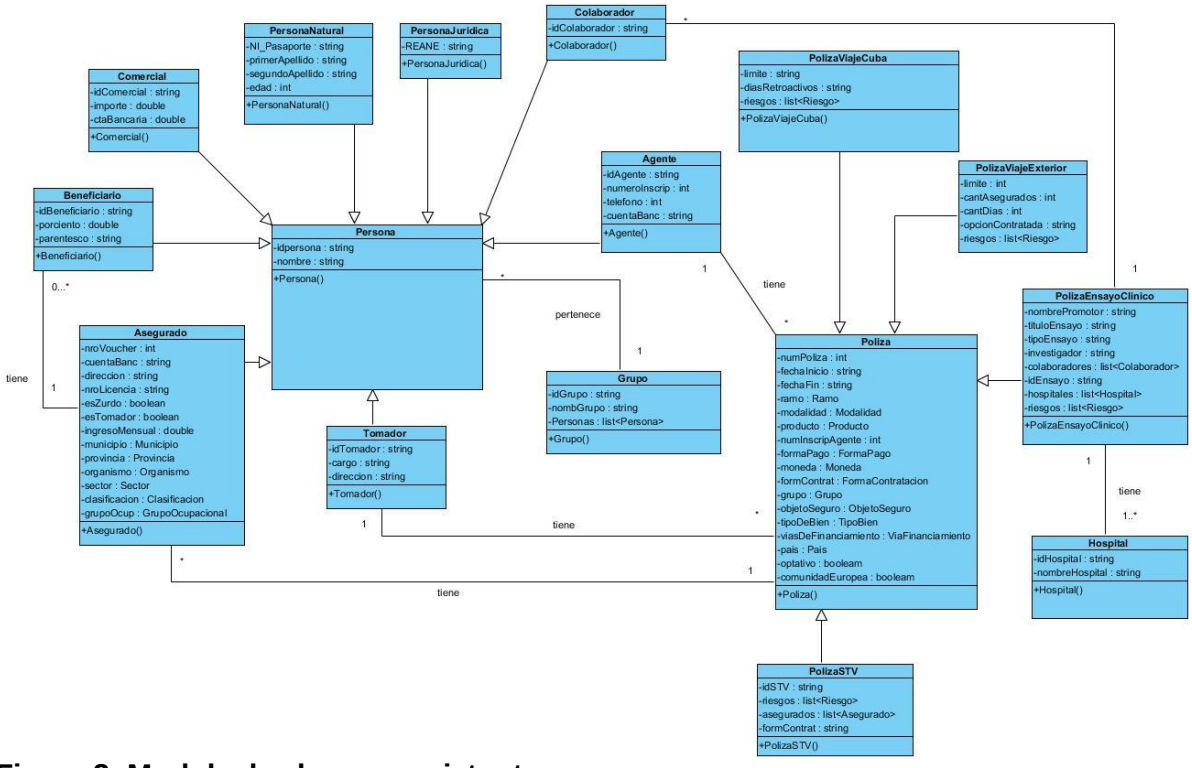


Figura 3: Modelo de clases persistentes.

2.5.3 Clases del diseño

2.5.3.1 Patrones de diseño

Los diagramas de clases del diseño necesitan el sustento de patrones de diseño para lograr una alta medida de calidad y precisión en su elaboración. Es por ello, que a continuación se estudian los patrones de diseño GRASP, debido a que son los más adecuados para las aplicaciones orientadas a objeto.

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. (Larman, 2002)

Patrón Experto:

Experto: se encarga de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. (Larman, 2002)

Beneficios del patrón Experto:

- ✓ Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. (Larman, 2002)
- ✓ El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase "sencillas" y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión. (Larman, 2002)

Patrón Creador:

Creador: se encarga de asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos (Larman, 2002):

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).

Beneficios del patrón Creador:

- ✓ Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Es probable que el acoplamiento no aumente, pues la clase creada tiende a ser visible a la clase creador**,** debido a las asociaciones actuales que nos llevaron a elegirla como el parámetro adecuado. (Larman, 2002)

Patrón Bajo acoplamiento:

Bajo acoplamiento: se encarga de asignar una responsabilidad para mantener bajo acoplamiento. (Larman, 2002)

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Mientras que una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente debido a que presentan los siguientes problemas (Larman, 2002):

- Los cambios de las clases afines ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.
- Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

Patrón Alta cohesión:

Alta cohesión: se encarga de asignar una responsabilidad de modo que la cohesión siga siendo alta. (Larman, 2002)

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Mientras que una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. Este último tipo de clase no es conveniente debido a que presenta los siguientes problemas (Larman, 2002):

- Son difíciles de comprender.
- Son difíciles de reutilizar.
- Son difíciles de conservar.
- Son delicadas: las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos. (Larman, 2002)

Beneficios del patrón Alta cohesión:

- ✓ Mejoran la claridad y la facilidad con que se entiende el diseño.
- ✓ Se simplifican el mantenimiento y las mejoras en funcionalidad.
- ✓ A menudo se genera un bajo acoplamiento.
- ✓ La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

Patrón Controlador:

Controlador: se encarga de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones (Larman, 2002):

- ➢ El "sistema" global (controlador de fachada).
- ➢ La empresa u organización global (controlador de fachada).
- ➢ Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- ➢ Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados "Manejador<NombreCasodeUso>" (controlador de casos de uso).

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema. (Larman, 2002)

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. (Larman, 2002)

Beneficios del patrón Controlador:

- ✓ Mayor potencial de los componentes reutilizables. Garantiza que la empresa o los procesos de dominio Sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Desde el punto de vista técnico, las responsabilidades del controlador podrían cumplirse en un objeto de interfaz, pero esto supone que el código del programa y la lógica relacionada con la realización de los procesos del dominio puro quedarían incrustados en los objetos interfaz o ventana. Un diseño de interfaz como controlador reduce la posibilidad de reutilizar la lógica de los procesos del dominio en aplicaciones futuras, por estar ligada a una interfaz determinada (por ejemplo, un objeto similar a una ventaja) que rara vez puede utilizarse en otras aplicaciones. En cambio, el hecho de delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio soporta la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras. (Larman, 2002)

2.5.3.2 Uso de los patrones de diseño

A continuación, se evidencia el uso de los patrones GRASP en el diseño de la solución:

Patrón Experto

Este patrón se utilizó en la clase Controladora que contiene toda la información necesaria para realizar las funcionalidades de obtener el listado de las pólizas, adicionar, eliminar, o modificar estas, guardar las pólizas y cargarlas.

Patrón Creador

El uso de este patrón ayudó a identificar a la clase Póliza como la clase responsable de la creación e instanciación de los objetos de la clase Asegurado. Lo cual promueve un bajo acoplamiento, mayor claridad, encapsulación y reutilización de estas clases.

Patrón Bajo acoplamiento:

El uso de este se evidencia al procurar mantener a las clases implementadas lo menos ligadas entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tuviera el mínimo impacto posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Uso del patrón Alta cohesión:

El uso de este patrón se pone en evidencia al asignarle a cada clase de la solución la responsabilidad de trabajar sobre una misma área, sin hacer complejo su comportamiento.

Uso del patrón Controlador:

Este patrón se pone en práctica al asignarle la responsabilidad a la clase PolizaControlller de realizar el manejo de los eventos del componente, realizando determinada funcionalidad en dependencia del evento. Este patrón posibilita que la lógica de negocios esté separada de la capa de presentación, por lo que facilita el aumento de la reutilización de código.

2.5.3.3 Diagrama de clases del diseño

A continuación, se presenta en la figura 4, el diagrama de clases del diseño para el requisito Registrar póliza, donde se refleja la arquitectura Cliente/Servidor, siendo representado el lado cliente con estereotipos web y el lado del servidor con los paquetes de servicio, web, dao y dominio. El resto de los diagramas de clases del diseño se encuentran en los anexos 1,2,3,4,5.

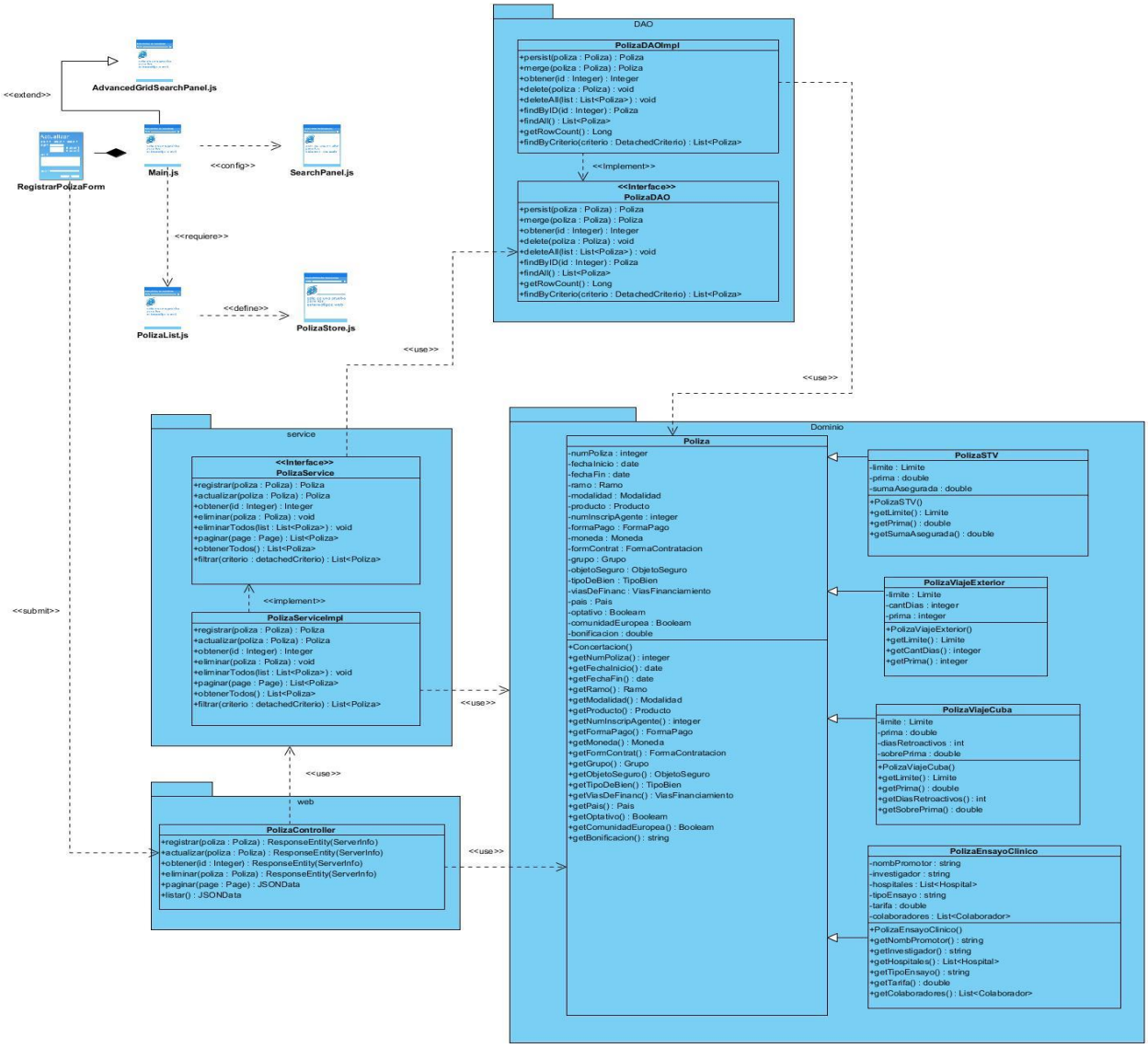


Figura 4: Diagrama de clases del diseño. Registrar póliza.

2.6 Validación del diseño

Para obtener software de calidad es preciso medir el proceso de desarrollo, cuantificar lo que se ha hecho y lo que falta por hacer, estimar el tamaño del programa, costos, tiempo de desarrollo y otros parámetros. La medición de este producto se realiza mediante las métricas, para caracterizar numéricamente los distintos aspectos del desarrollo del software.

Las métricas de software tienen un papel decisivo en la obtención de un producto de alta calidad, porque determinan mediante estadísticas basadas en la experiencia, el avance del software y el cumplimiento de parámetros requeridos. Siempre habrá elementos cualitativos para la creación de software. El problema estriba en que la valoración cualitativa puede no ser suficiente. Un ingeniero del software necesita criterios objetivos para guiarse en el diseño de datos, de la arquitectura, de las

interfaces y de los componentes. El verificador necesita una referencia cuantitativa que le ayude en la selección de los casos de prueba y de sus objetivos. Las métricas técnicas facilitan una base para que el análisis, diseño, codificación y prueba puedan ser conducidas más objetivamente y valoradas más cuantitativamente." (Pressman, 2002)

Las métricas empleadas para medir la calidad del diseño Orientado a Objeto están ajustadas a las características de la Programación Orientada a Objeto, por lo tanto, se basan en el encapsulamiento, acoplamiento, cohesión, complejidad, polimorfismo y reutilización. Para medir el diseño existen numerosas métricas, pero entre las más referenciadas se pueden encontrar la familia de métricas de propuestas por Chidamber y Kemerer y las de Lorenz y Kidd.

Teniendo en cuenta la relevancia de los autores antes mencionados, y en aras de determinar el grado de calidad y fiabilidad del diseño se decide aplicar la métrica Acoplamiento entre Objetos (CBO) de Chidamber y Kemerer y la métrica Número de Métodos de Clase (NCM) de Lorenz y kidd, debido a que miden los atributos de calidad responsabilidad, complejidad, reutilización, acoplamiento, mantenimiento, cantidad de pruebas y reutilización, brindando una alta medida de la fiabilidad y la calidad del diseño realizado.

NCM: se basa en número total de métodos a nivel de clase. Un método de clase es un método que es global para sus instancias. El número de métodos de la clase puede indicar la cantidad de elementos comunes que se manejan para todas las instancias. Este número generalmente debe ser relativamente pequeño en comparación con el número de métodos de instancia (Lorenz, y otros, 1994).

CBO: se basa en el número de clases a las que ella está relacionada, sin tener en cuenta las relaciones por herencia, y mide la complejidad de la misma. El CBO alto no es deseado porque puede ser perjudicial para el diseño y mide la posible reutilización (mientras más independiente es una clase más fácil es de reutilizar). También un alto valor del mismo reduce el encapsulamiento y da medida de la complejidad de pruebas. (Chidamber, y otros, 1994)

2.6.1 Resultados de la aplicación de la métrica NCM

La Figura 5 muestra los resultados obtenidos de la aplicación de la métrica NCM al diseño. Estos resultados se agrupan en los intervalos representados en la gráfica. El gráfico refleja que la mayoría de las clases tienen de uno a cinco procedimientos.

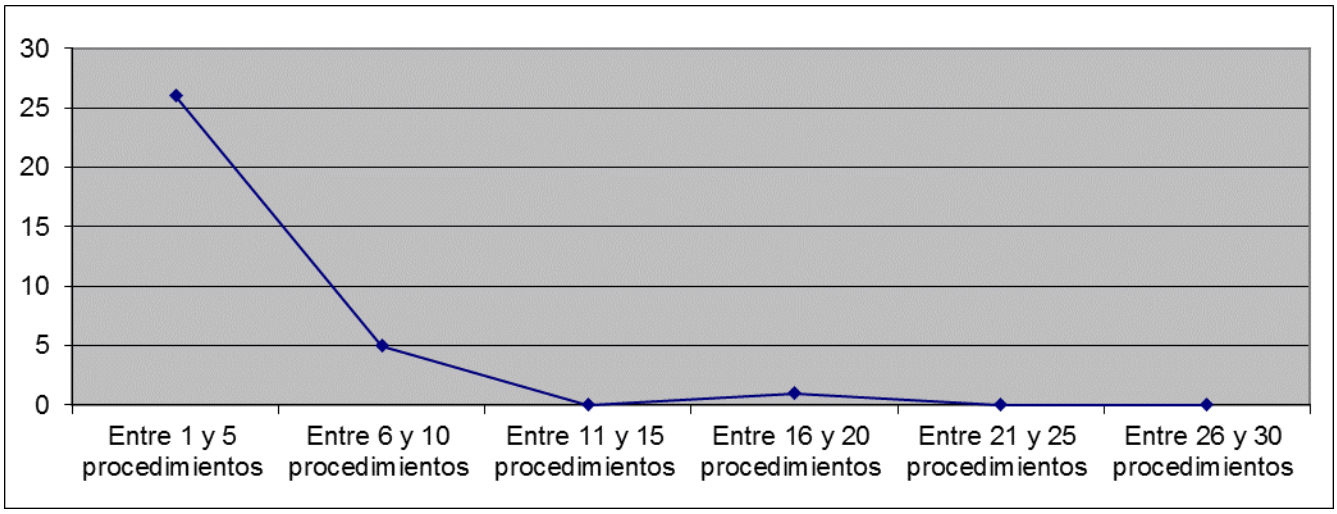


Figura 5: Representación de la evaluación de la métrica NCM

Este resultado demuestra que el funcionamiento general del sistema está distribuido equitativamente entre la mayoría de las clases, aunque es necesario analizar las que contienen la mayor cantidad de procedimientos para tratar de restarle funcionalidades y distribuirlas entre las demás clases para evitar que alguna sea demasiado crítica y que en caso de fallo sea menor el número de funcionalidades que queden fuera de servicios.

En la Figura 6 se observa la representación de la incidencia de los resultados de la evaluación de la métrica NCM en el atributo Responsabilidad. La gráfica muestra un resultado satisfactorio ya que el 75 % de las clases poseen una baja responsabilidad.

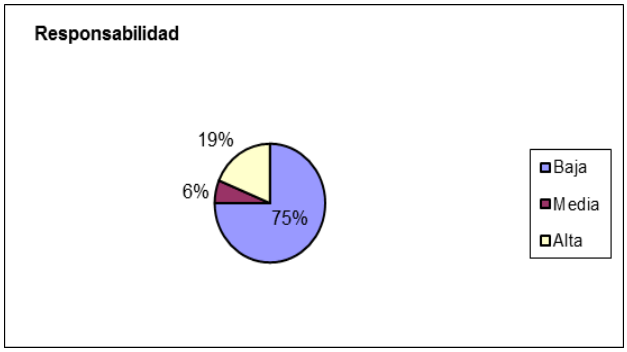


Figura 6: Representación de la incidencia de la métrica NCM en el atributo Responsabilidad

La Figura 7 muestra el resultado de la incidencia de la métrica NCM en el atributo Complejidad de Implementación (izquierda). Este gráfico muestra un resultado satisfactorio, pues el 75 % de las clases poseen una baja complejidad de implementación, característica que permite mejorar el mantenimiento y soporte de estas clases. Además, la figura muestra en su parte derecha el resultado de la incidencia de la métrica NCM en el atributo Reutilización. Este gráfico muestra que el diseño de

la herramienta tiene un grado aceptable pues solo el 9 % del total de las clases posee una baja reutilización.

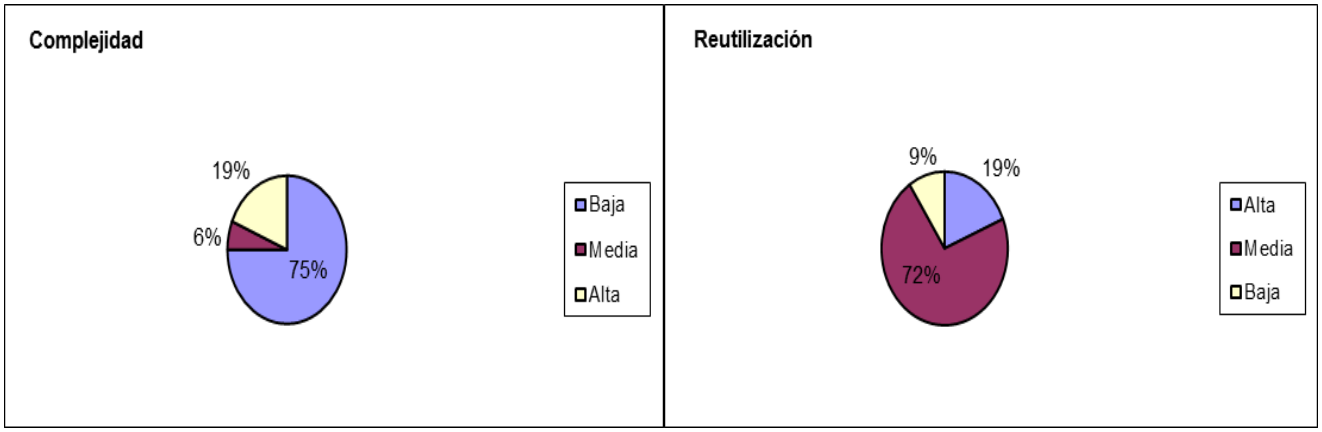


Figura 7: Resultados evaluación de la métrica NCM en: Complejidad de implementación; Reutilización.

Teniendo en cuenta los resultados de la aplicación de la métrica NCM al diseño, se puede señalar que en general tiene una calidad aceptable, pues el mayor por ciento del total de las clases posee baja responsabilidad, baja complejidad de implementación y mediana reutilización. Los instrumentos y las tablas de resultados del instrumento de medición de la métrica TOC se pueden observar en el Anexo 7.

2.6.2 Resultados de la aplicación de la métrica CBO

La Figura 8 muestra los resultados obtenidos en el instrumento de evaluación de la métrica CBO en porcientos agrupados en los intervalos definidos. El gráfico refleja que el 69 % de las clases tienen entre 1 y ninguna dependencia con otra clase, demostrando que la mayoría de las clases presentan niveles aceptables de calidad.

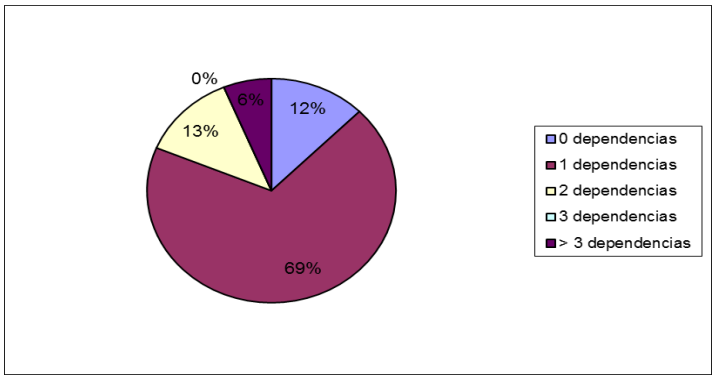


Figura 8: Resultados obtenidos en los intervalos definidos según la métrica CBO.

La Figura 9 muestra la representación de la incidencia de los resultados de la evaluación de la métrica CBO en los atributos: Acoplamiento (izquierda) y Complejidad de mantenimiento (derecha). En el gráfico de la izquierda se evidencia un diseño eficiente al quedar reflejado que el 69 % de las clases presenta bajo acoplamiento, mientras que el 12 % no presenta dependencias con otras clases, lo que aumenta el grado de reutilización del componente. En el gráfico de la derecha el resultado es favorable ya que el 81 % de las clases presenta baja complejidad de mantenimiento, dándole respaldo a una buena calidad de la arquitectura.

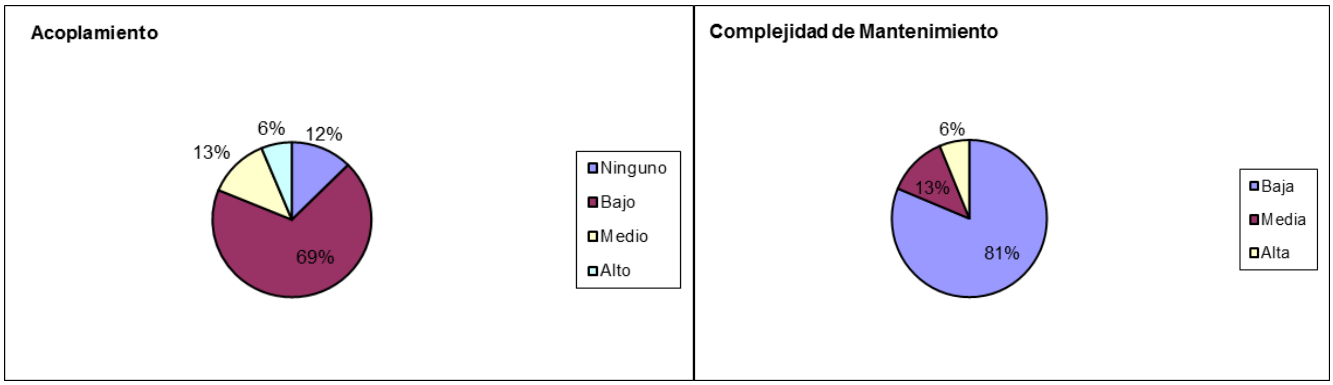


Figura 9: Resultados evaluación de la métrica CBO para: Acoplamiento; Complejidad de mantenimiento

La Figura 10 muestra la representación de la incidencia de la métrica CBO en los atributos: Cantidad de pruebas (izquierda) y Reutilización (derecha). El gráfico de la izquierda demuestra que la mayor cantidad de las clases no necesita una cantidad elevada de pruebas según las dependencias entre las clases. Por su parte el gráfico de la derecha muestra que el diseño tiene un 81 % de reutilización de sus clases, constituyendo un factor importante en el desarrollo del software.

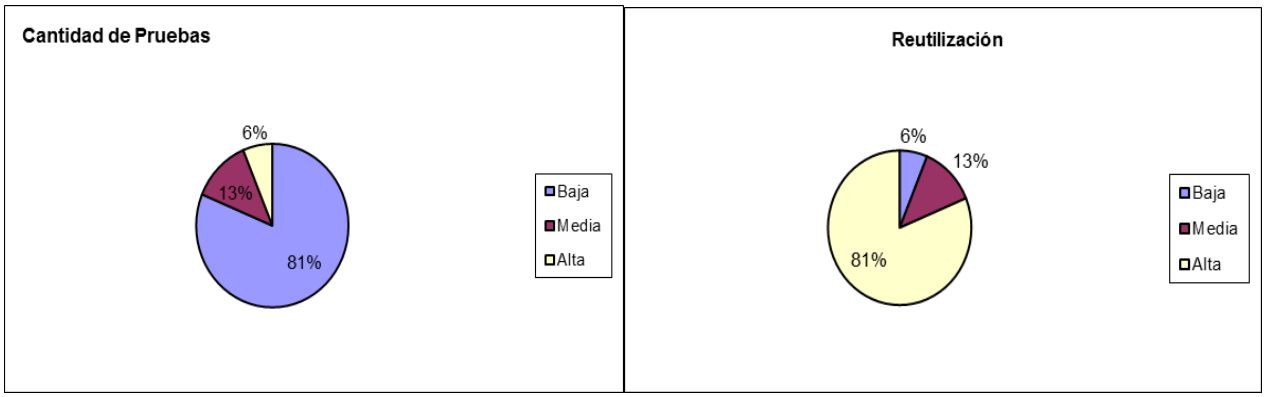


Figura 10: Resultados evaluación de la métrica CBO para: Cantidad de pruebas; Reutilización

Analizando los resultados de la evaluación del instrumento de medición de la métrica CBO, se puede concluir que el diseño posee una calidad aceptable, debido a que el 81 % de las clases de la solución tienen menos de 2 dependencias con otras clases. También se refleja en los resultados que en el 81 % el acoplamiento entre clases es mínimo; que la complejidad de mantenimiento y la cantidad de pruebas son bajas en un 81 %; y que la reutilización se comporta en un valor elevado del 81 %. Este resultado demuestra que los atributos de calidad se encuentran en niveles satisfactorios. Los instrumentos y las tablas de resultados del instrumento de medición de la métrica CBO se pueden observar en el Anexo 8.

2.7 Conclusiones del capítulo

La modelación del proceso de negocio permitió comprender como se realiza la concertación de pólizas en la ESEN y refinar el modelo conceptual. Los artefactos generados en el diseño permitieron definir la estructura de la base de datos, así como las clases y sus relaciones para realizar la implementación del componente acorde a los requisitos. Por otro lado, la validación del diseño mediante el uso de métricas verificó el cumplimiento de los atributos de calidad de los artefactos generados.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

Este capítulo comprende el proceso de implementación del componente para el que se tiene en cuenta los estándares de codificación definidos. Se representan además las pruebas de software realizadas sobre la aplicación en aras de detectar no conformidades y dar solución a las mismas.

3.2 Estándares de código

A continuación, se describe el estándar de código utilizado para el desarrollo del componente Concertación de pólizas de seguro. (Proyecto ESEN, 2015)

3.2.1 Convención de nomenclatura

Variables: se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra será con mayúscula.
Ejemplo: formaPago

Clases: se rigen por la nomenclatura Dromedarian y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.
Ejemplo: AseguradoPoliza

Funciones: se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa.
Ejemplo: setFormaPago(String formaPago)

Controladoras: mismo nombre de la clase que representa seguido de la palabra “Controller”.
Ejemplo: PolizaController

Servicios: mismo nombre de la clase que representa seguido de la palabra “Service”.
Ejemplo: PolizaService

3.2.2 Documentación

Todos los archivos deben de tener la documentación asociada al mismo. Para esto debe de cumplirse con la documentación de tipo comentario en aras de facilitar el trabajo a la hora del completamiento de código que brinda los entornos de desarrollo actuales.

3.2.3 Prototipo funcional del componente

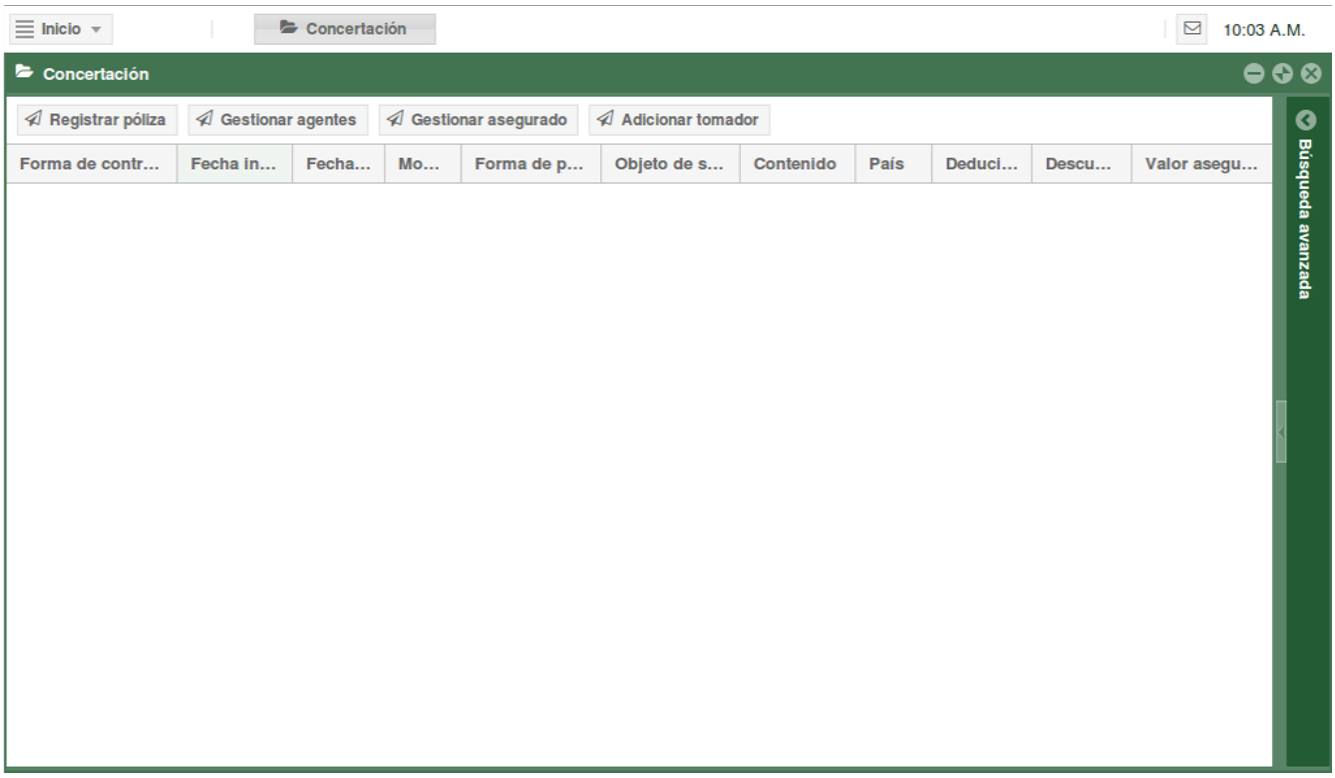


Figura 10: Prototipo funcional del componente Concertación.

3.2.4 Mapa de navegación

En la figura 11, se ilustra el mapa de navegación para acceder al componente concertación dentro del sistema.

Figura 11: Mapa de Navegación

En la figura 12, se ilustra el mapa de navegación del componente concertación para acceder a las opciones del mismo.

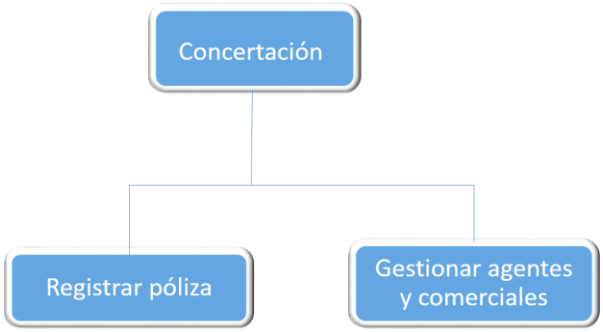


Figura 12: Mapa de Navegación del componente.

3.3 Pruebas

Como parte del proceso de desarrollo, las distintas disciplinas de pruebas añaden valor al producto que se maneja, debido a que el desarrollo de sistemas de software lleva implícito una serie de actividades en las cuales, los fallos y los errores son frecuentes desde el momento inicial, de ahí que esta tiene como objetivo específico encontrarlos.

“Las pruebas de software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y la codificación. Los procesos que verifican y revelan la calidad de un producto de software, son utilizados para identificar los posibles fallos de la implementación, calidad o la usabilidad de una aplicación informática” (Pressman, 2002).

Por otro lado, (Sommerville ,2005) sobre esta parte del proceso de desarrollo de software apunta: “Dentro de los objetivos del proceso de pruebas están: demostrar al desarrollador y al cliente que el software satisface sus requisitos y para descubrir defectos en el software, que el comportamiento de este es incorrecto, no deseable o no cumple su especificación. El primer objetivo conduce a las pruebas de validación, en las que se espera que el sistema funcione correctamente usando un conjunto determinado de casos de prueba que reflejan el uso esperado de aquel. El segundo objetivo conduce a la prueba de defectos, en los que los casos de prueba se diseñan para exponer los defectos. Los casos de prueba pueden ser deliberadamente oscuros y no necesitan reflejar cómo se utiliza normalmente el sistema.”

A partir de los criterios anteriores se puede concluir que el objetivo de las pruebas es detectar un error no descubierto hasta entonces, y estas tienen éxito si lo descubren.

El software debe probarse desde dos perspectivas diferentes:

- Lógica interna del programa: se comprueba utilizando técnicas de diseño de casos de pruebas de "caja blanca".
- Los requisitos del software: se comprueban utilizando técnicas de diseño de casos de prueba de "caja negra".

3.3.1 Pruebas de caja blanca

Las pruebas de caja blanca comprueban los componentes internos. Comprueba los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado.

Este método de casos de prueba usa los detalles procedimentales del programa. Se busca obtener casos de prueba que:

- ✓ Garanticen que se ejecuta por lo menos una vez todos los caminos independientes de cada módulo.
- ✓ Verificar las decisiones lógicas (V/F).
- ✓ Ejecutar las estructuras internas de datos para asegurar su validez.

Dentro del método de caja blanca se incluyen varias pruebas tales como:

- ✓ **La prueba del camino básico:** permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

- ✓ **La prueba de condición:** es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

- ✓ **La prueba de flujo de datos:** se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- ✓ **La prueba de bucles:** es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

Pruebas de caja blanca aplicadas al componente

Para verificar el correcto funcionamiento de los procedimientos del componente al efectuarse el proceso de registro de agentes y comerciales se realizaron pruebas funcionales aplicando la técnica de partición de equivalencia.

A continuación, se presenta el caso de prueba asociado a la técnica partición de equivalencia.

La figura muestra las sentencias de código del método para registrar un asegurado, las cuales se encuentran enumeradas y posteriormente se encuentra el grafo de flujo correspondiente a este método.

```
public ResponseEntity<ServerInfo> registrar(@Valid @RequestBody String asegurado, BindingResult result) throws FormValidationError{  //1
    if(result.hasErrors()) {  //2
        throw new FormValidationError(result.getAllErrors()); //3
    } //4
    else { //5
        JSONObject jsonObject = new JSONObject(asegurado);
        Municipio municipio = new Municipio(jsonObject.getInt("municipio"));
        Provincia provincia = new Provincia(jsonObject.getInt("provincia"));
        Organismo organismo = new Organismo(jsonObject.getInt("organismo"));
        Sector sector = new Sector(jsonObject.getInt("sector"));
        ClasificacionAsegurado  clasificacionAsegurado = new ClasificacionAsegurado(jsonObject.getInt("clasificacionAsegurado"));
        PersonaNatural natural = new PersonaNatural();
        natural.setNombre(jsonObject.getString("nombre"));
        natural.setSexo(new Sexo(1));
        natural.setNiPasaporte("1");
        natural.setPrimerApellido(jsonObject.getString("primerApellido"));
        natural.setSegundoApellido(jsonObject.getString("segundoApellido"));
        natural.setEdad(jsonObject.getInt("edad"));
        BigInteger noVoucher = new BigInteger(jsonObject.getString("noVoucher"));
        String direccion = jsonObject.getString("direccion");
        BigInteger ctaBancaria = new BigInteger(jsonObject.getString("ctaBancaria"));
        Asegurado aseg = new Asegurado();
        aseg.setPersona(natural);
        aseg.setMunicipio(municipio);
        aseg.setProvincia(provincia);
        aseg.setOrganismo(organismo);
        aseg.setSector(sector);
        aseg.setNoVoucher(noVoucher);
        aseg.setDireccion(direccion);
        aseg.setCtaBancaria(ctaBancaria);
        aseg.setClasificacionAsegurado(clasificacionAsegurado);
        this.aseguradoConcService.registrar(aseg);  //6
        return this.success(this.getMensaje("base.operacion.satisfactoria")); //7
    } //8
}  //9
```

Figura 14: Método para registrar los asegurados.

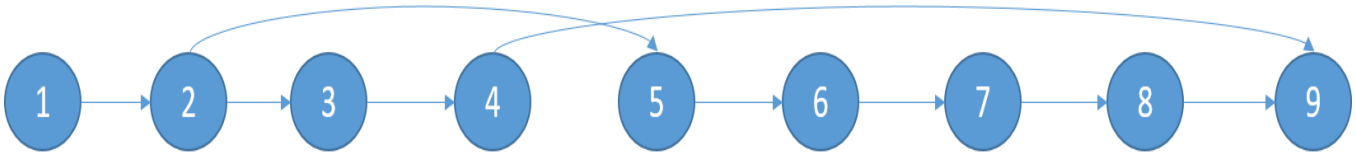


Figura 15: Grafo de flujo asociado al método.

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, la cual garantiza la menor cantidad de casos de pruebas a realizar, para que se ejecute al menos una vez cada sentencia de código. El cálculo de la complejidad ciclomática es necesario efectuarlo mediante tres vías o fórmulas, se debe utilizar el mismo grafo en cada caso:

Fórmulas	**Nomenclaturas**	**Resultados**
(G) = (A – N) + 2	A: cantidad total de aristas del grafo. N: cantidad total de nodos del grafo.	(G) = (9 - 9) + 2 (G) = 2
(G) = P+1	P: cantidad total de nodos predicados. Un nodo es predicado cuando parten del mismo 2 o más aristas.	(G) = 2 + 1 (G) = 3
(G) = R	R: cantidad total de regiones existentes en el grafo, se incluye el área exterior del grafo, como una región más.	(G) = 2

Tabla 6: Fórmulas para el cálculo de la complejidad ciclomática.

Se especifican los caminos básicos que puede tomar el algoritmo durante su ejecución. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

Número	Caminos básicos
1	1-2-3-4-9
2	1-2-5-6-7-8-9

Tabla 7: Caminos básicos del flujo.

Luego de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento. Para realizarlos es necesario cumplir con las siguientes exigencias:

Descripción: se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: se muestran los parámetros que entran al procedimiento.

Resultados esperados: se expone el resultado que se espera que devuelva el procedimiento.

Resultados: se muestra el resultado obtenido.

Seguidamente se presentan los casos de pruebas de caja blanca generados

Camino básico #1: 1-2-3-4-9	
Descripción	Se validan los datos del asegurado y se encuentran datos incorrectos.
Condición de ejecución	Se debe tener el resultado (result) de la validación de los datos y los datos del asegurado.
Entrada	result.hasErrors() == true
Resultado esperado	Mensaje de alerta de los errores encontrados
Resultado obtenido	Satisfactorio.

Tabla 8: Caso de prueba para el camino básico # 1.

Camino básico #2: 1-2-5-6-7-8-9	
Descripción	Se validan los datos del asegurado, se encuentran datos correctos, se registra el asegurado y se muestra un mensaje de operación satisfactoria.
Condición de ejecución	Se debe tener el resultado (result) de la validación de los datos y los datos del asegurado.
Entrada	result.hasErrors() == false
Resultado esperado	Se registra un nuevo asegurado.
Resultado obtenido	Satisfactorio.

Tabla 9: Caso de prueba para el camino básico # 2

3.3.2 Pruebas de caja negra

Las pruebas de caja negra comprueban las funcionalidades sin tener en cuenta la estructura interna. Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, a través de los casos de prueba se demuestra que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Este tipo de pruebas permite encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores de rendimiento.

Dentro de la prueba de caja negra se incluyen varias técnicas de pruebas tales como:

- ✓ **Partición de equivalencia:** técnica que divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ **Análisis de valores límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ **Grafos de causa-efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Pruebas de caja negra aplicadas al componente

Para verificar el correcto funcionamiento de la solución para la recuperación de la información al efectuarse el proceso de registro de agentes y comerciales se realizaron pruebas funcionales aplicando los métodos de caja negra, específicamente la técnica de partición de equivalencia A continuación, se presenta el caso de prueba asociado a la técnica partición de equivalencia.

Condiciones de ejecución

1. Se debe identificar y autenticar ante el sistema y tener los permisos para ejecutar esta acción.
2. Se debe ir a **Inicio/ Desarrollo/ Concertación/Gestionar agentes y comerciales**
3. Se selecciona la opción **Adicionar agente natural.**

Requisito Adicionar agente natural.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas**	**Flujo del escenario**
1: Adicionar agente natural	El objetivo de este requisito es registrar agentes naturales.	EP1.1: Adicionar agente natural.	✓ Se introducen todos los datos correctamente. ✓ Se presiona el botón **Adicionar** para registrar el agente natural en el sistema. ✓ El sistema muestra una ventana de información con el texto: "*El agente natural ha sido adicionado satisfactoriamente*.". ✓ Se presiona el botón **Aceptar** de la ventana de información.
		EP1.2: Adicionar un agente natural con datos obligatorios	✓ Se dejan campos obligatorios vacíos. ✓ Se presiona el botón **Adicionar** para registrar el agente natural. ✓ El sistema muestra los campos vacíos en rojo con el mensaje:

		vacíos.	"*Este campo es obligatorio*.".
		EP1.3: Adicionar un agente natural con datos repetidos.	✓ Se introducen datos iguales a los de otro agente natural registrado en el sistema. ✓ Se presiona el botón **Adicionar** para registrar el agente natural. ✓ El sistema muestra el mensaje: "*Estos valores ya existen*".
		EP14: Cancelar.	✓ Se introducen los datos. ✓ Se presiona el botón **Cancelar**. ✓ Se limpian los campos.

Tabla 3: Diseño de caso de prueba del requisito Adicionar agente natural.

No.	**Nombre del campo**	**Clasificación**	**Puede ser nulo**	**Descripción**
1	Descripción	Campo de texto y números.	No	Alfanumérico
2	Caja	Combobox	No	Solo permite seleccionar los elementos que están en el Combobox.

Tabla 4: Descripción de variables para el requisito Adicionar agente natural.

Id del escenario	Escenario	Descripción	Respuesta del sistema
EP 1.1	Adicionar un agente natural entrando datos válidos.	Se llenan todos los campos del formulario con datos válidos.	El sistema muestra una ventana de información con el texto: "El agente natural ha sido adicionado satisfactoriamente.". Se acepta el mensaje de información y se limpian todos los campos.
EP 1.2	Adicionar agente natural entrando datos inválidos.	Se llenan los campos del formulario entrando datos inválidos. Ejemplo: - nombre: 2dfe44 - apellidos: 123-2 - edad: yi1	El sistema muestra los campos con datos inválidos en rojo con el mensaje: "*Datos no válidos*.".

Tabla 5: Juego de datos a probar del requisito Adicionar agente natural.

3.3.3 Resultados de las pruebas realizadas

Luego de aplicados los métodos de prueba a la solución desarrollada, se detectaron un grupo de no conformidades, las cuales se ilustran a continuación en la tabla 6:

Requisitos	Cantidad de no Conformidades	Resueltas
Adicionar asegurado	5	5
Modificar asegurado	2	2
Eliminar asegurado	1	1

Listar asegurados	1	1
Buscar asegurados	0	0
Adicionar beneficiario	2	2
Modificar beneficiario	0	0
Eliminar beneficiario	1	1
Listar beneficiarios	0	0
Buscar beneficiarios	0	0
Registrar agentes y comerciales	6	6
Modificar agentes y comerciales	2	2
Eliminar agentes y comerciales	0	0
Listar agentes y comerciales	1	1
Buscar agentes y comerciales	0	0
Registrar tomador	0	0
Buscar tomador	0	0
Asociar asegurado a la póliza	0	0
Asociar beneficiario a asegurado	1	1
Asociar tomador a la póliza	0	0
Calcular prima	2	2
Registrar póliza	10	10
Listar pólizas	2	2

Tabla 6 Resultados de las pruebas por requisitos funcionales

Para resolver las no conformidades se realizaron 2 iteraciones, donde quedaron resueltas el 100% de las mismas, lo cual se ilustra en la figura 16.

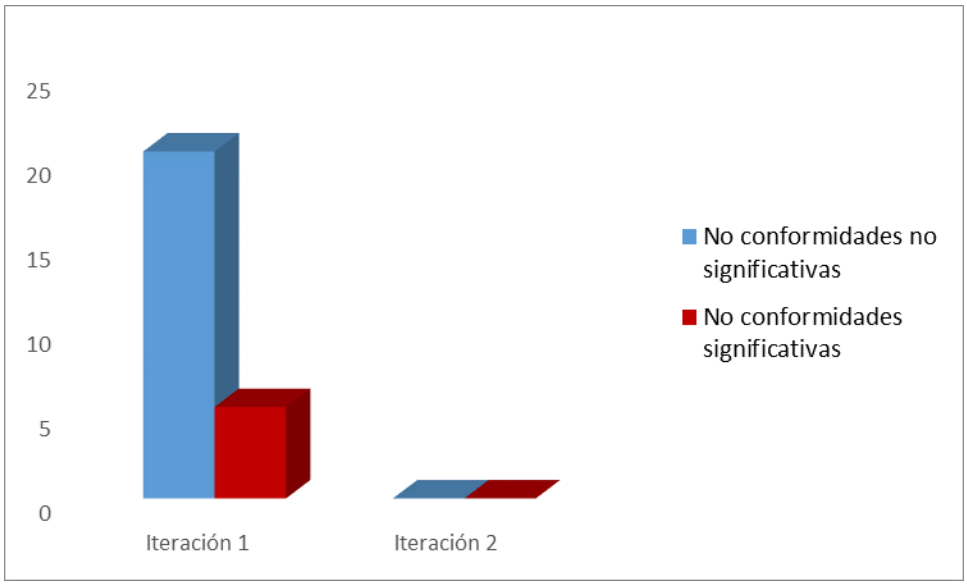


Figura 16: Gráfica de No conformidades.

3.4 Conclusiones del capítulo

La definición de los estándares de codificación y la convención de nomenclaturas permitieron implementar el componente con una baja complejidad de mantenimiento. La realización de las pruebas de caja negra y caja blanca corroboraron el correcto funcionamiento de la solución propuesta, permitiendo detectar las no conformidades y resolverlas, cumpliendo así con los requisitos pactados con el cliente.

CONCLUSIONES GENERALES

Como conclusiones generales de la investigación se muestran alcanzados los objetivos propuestos satisfactoriamente:

- ✓ Elaborar el marco teórico de la investigación que permita comprender el proceso de concertación de pólizas de seguro, así como el entorno donde se desarrollará la solución. Para este objetivo se realizó un estudio del proceso de concertación de pólizas de seguro, donde se obtuvo qué actividades lo componen, cómo se ejecutan y que artefactos se generan de dicho proceso. También se realizó un estudio de la metodología, lo cual permitió identificar las nuevas actividades y tareas a incorporar según las características del proyecto. Además, se realizó una investigación de los principales sistemas de gestión de pólizas, lo cual sirvió como soporte para el desarrollo de la propuesta de solución.
- ✓ Diseñar el componente Concertación basado en los requisitos previamente identificados. Para darle cumplimiento a este objetivo, en el Capítulo 2, se realizó la descripción del proceso de negocio, lo cual permitió comprender como se realiza la ejecución del mismo y con ello generar los artefactos necesarios para el correcto diseño de la solución.
- ✓ Validar el diseño del componente Concertación mediante el uso de métricas. Dicho objetivo se cumplió a través del estudio de las diferentes métricas de validación del diseño, donde se definieron como métricas más adecuadas NCM y CBO, corroborando de esta forma que el diseño cumple con la calidad requerida.
- ✓ Implementar el componente Concertación teniendo en cuenta el diseño realizado. Para darle cumplimiento a este objetivo, se definieron los estándares de codificación y la convención de nomencladores, permitiendo desarrollar el componente de manera que sea de fácil mantenimiento y entendible al usuario.
- ✓ Verificar y validar el componente obtenido mediante la realización de pruebas de software. Para este objetivo se realizó un estudio de los tipos de pruebas de caja blanca y caja negra, determinándose realizar la prueba del camino básico para medir la complejidad lógica de la aplicación como parte de las pruebas de caja blanca y la técnica de partición de equivalencia en las pruebas de caja negra, lo cual permitió identificar las no conformidades presentes en el componente y resolverlas.

Por lo anteriormente descrito se concluye el cumplimiento del objetivo general de la investigación desarrollar el componente Concertación, cumpliendo con los requisitos identificados, de manera que se registre toda la información del proceso de Concertación de pólizas de seguros y permita el acceso a la misma.

RECOMENDACIONES

Se considera que de forma general se cumplieron los objetivos planteados, esta propuesta constituye la primera fase del proceso de gestión de pólizas de seguro por lo que se propone realizar la integración con los demás componentes para que el mismo obtenga una información actualizada sobre las pólizas.

REFERENCIAS BIBLIOGRÁFICAS

Aguirre Pérez, Yurisel. 2012. "Sistema para la detección de plagio y validación de estructura en los documentos científicos emitidos en el Centro de Información Científico Técnica (CICT) del ISMMM" en Revista Caribeña de Ciencias Sociales, Disponible a: http://caribeña.eumed.net/sistema-para-la-deteccion-de-plagio-y-validacion-de-estructura-en-los-documentos-cientificos-emitidos-en-el-centro-de-informacion-cientifico-tecnica-cict-del-ismmm/

Arias, Á. 2014. Aprende a Programar con Ajax. IT Campus Academy. ISBN 978-1-4991-6894-5.

Arias, Á. M. i Durango, A. 2016. Curso de Desarrollo Web: 2ª Edición. IT Campus Academy. ISBN 978-1-5308-0869-4.

ARQUITECTURA CLIENTE SERVIDOR. 2012. A: Scribd [en línea]. [Consulta: 19 abril 2016]. Disponible a: https://es.scribd.com/doc/49374795/ARQUITECTURA-CLIENTE-SERVIDOR.

Portal del IDE Java de Código Abierto. 2016. [en línea]. [Consulta: 19 abril 2016]. Disponible a: https://netbeans.org/index_es.html.

BRG. 2016. Defining «Business Rule»**.** A: Business Rules Group [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.businessrulesgroup.org/defnbrg.shtml.

Chidamber, Shyam R. y Kemerer, Chris F. 1994. A Metrics Suite for Object Oriented Design. s.l.: IEEE Transactions on Software Engineering., 1994. vol. 20, no. 6.

Concertación. 2015. Significado de concertación diccionario**.** A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://es.thefreedictionary.com/concertaci%C3%B3n.

DAO support. A: [en línea]. [Consulta: 19 abril 2016]. Disponible a: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/dao.html.

DLE: persona. 2015. Diccionario de la lengua española. Edición del Tricentenario**.** A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://dle.rae.es/?id=SjUlL8Z.

DLE: váucher. 2015. Diccionario de la lengua española. Edición del Tricentenario. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://dle.rae.es/?id=bPiXazx.

Bascón Pantoja, Ernesto. 2004 *El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing* [en línea]. [Consulta: 8 de abril del 2016]. Disponible a: https://www.academia.edu/5217432/El_patr%C3%B3n_de_dise%C3%B1o_Modelo-Vista-Controlador_MVC_y_su_implementaci%C3%B3n_en_Java_Swing.

ESEN. 2016. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.esen.cu.

ESEN - Seguro de Bienes agrícolas. 2016. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.esen.cu/index.php?page=bagr.

ESEN - Seguro de Bienes pecuarios. 2016. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.esen.cu/index.php?page=bpec.

ESEN - Seguro de Responsabilidad Civil. 2016. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.esen.cu/index.php?page=jcivil.

ESEN - Seguro de Vehículos de Transporte Terrestre. 2016. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.esen.cu/index.php?page=jauto.

ESEN - Seguro de Viajes. 2016. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.esen.cu/index.php?page=nvext.

ESEN - Seguro Temporario de Vida. 2016. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.esen.cu/index.php?page=nvida.

El patrón MVC (Symfony 1.2, la guía definitiva). A: [en línea]. [Consulta: 19 abril 2016]. Disponible a: http://librosweb.es/libro/symfony_1_2/capitulo_2/el_patron_mvc.html.

Ext JS - JavaScript. MVC/MVVM framework for cross-platform web apps | Sencha. A: [en línea]. [Consulta: 5 febrero del 2016]. Disponible a: https://www.sencha.com/products/extjs/#overview.

Gaceta Oficial No. 005 / 2009. EXTRAORDINARIA - Págs. (27 - 38) - Decreto263delContratodeSeguro.pdf [en línea]. [Consulta: 18 abril 2016 a]. Disponible a: http://www.esicuba.cu/Documentos/Decreto263delContratodeSeguro.pdf.

García, Rosa María Mato. 2005. *Sistemas de Bases de Datos.* [ed.] José Quesada Pantoja. La Habana : Pueblo y Educación, 2005. págs. 61-63. 959-13-1273-3.

GIS. 2016. programa de gestión de agencias, corredurías y agentes de seguros. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.infonetsoftware.com/infonetgis/quees.htm.

Gómez, O. El seguro de vida, un instrumento financiero indispensable para el bienestar económico familiar. Bogotá: Colegio de Estudios Superiores de Administración: s.n., 2001.

IntelliJ IDEA Features. *JetBrains* [online], [Accessed 29 June 2016]. Available from: https://www.jetbrains.com/idea/features/

InterHelper. 2016. Software de Seguros**.** A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.interhelper.net/.

Java. A: [en línea]. [Consulta: 19 abril 2016]. Disponible a: https://www.java.com/es/download/faq/whatis_java.xml.

Hernando Bravo Reyes, Juan. 2011. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://revistas.lasalle.edu.co/index.php/gs/article/viewFile/277/211.

Larman, Craig. 2002. UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. [ed.] Pearson Education. [trad.] Begoña Moros Valle. Segunda. Madrid : Prentice Hall, 2002. págs. 191-210. 8420534382, 9788420534381.

Lic. Noemí Benítez Rojas, Superintendente de Seguros. 2011. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.mfp.cu/docs/Historia%20del%20Seguro%20en%20Cuba.pdf.

Lorenz, M. y Kidd, J. 1994. Object Oriented Metrics. Englewood, New Jersey: Prentice Hall, 1994.

Mohedano, J. Saiz, J.M. i Román, P.S. 2012. Iniciación a Javascript. Ministerio de Educación. ISBN 978-84-369-5433-3.

NEKÜL. 2013. Soft de seguros gratuito para PAS. [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://nekul.com.ar/nekulweb/index.html.

NOA Gestión de seguros v5.0. 2016. A: ABCdatos [en línea]. 2016. [Consulta: 18 abril 2016]. Disponible a: http://www.abcdatos.com/programa/gestion-polizas-seguro.html.

Pressman, Roger S. 2002. Ingeniería de Software, un enfoque práctico. Quinta edición. McGraw-Hill, 2003. ISBN 84-481-3214-9.

Proyecto ESEN. 2015. Estándar de codificación para Java.

SearchData Center. 2015. ¿Qué es Base de datos relacional? - Definición en WhatIs.com. [en línea]. [Consulta: 11 mayo 2016]. Disponible a: http://searchdatacenter.techtarget.com/es/definicion/Base-de-datos-relacional.

Rafael Martinez. 2010. PostgreSQL. A: PostgreSQL-es [en línea]. [Consulta: 22 abril 2016]. Disponible a: http://www.postgresql.org.es/sobre_postgresql.

Rios, S. 2015. JSF 2 + Hibernate 4 + Spring 4: PrimeFaces 5 with JAX-WS y EJB'S. Sergio Rios.

Román, I.R. i Cosín, J.J.D., 2007. Técnicas cuantitativas para la gestión en la ingeniería del software. Netbiblo. ISBN 978-84-9745-204-5.

seQurnet v2.0. 2015. A: [en línea]. [Consulta: 18 abril 2016]. Disponible a: http://www.abcdatos.com/programa/mediadores-seguros.html.

Sommerville, I. i Galipienso, 2005. M.I.A. Ingeniería del software. Pearson Educación. ISBN 978-84-7829-074-1.

Sommerville, Ian. 2005. Ingeniería de Software, Séptima edición. Madrid: Pearson Education SA. ISBN 978-84-7829-074-1.

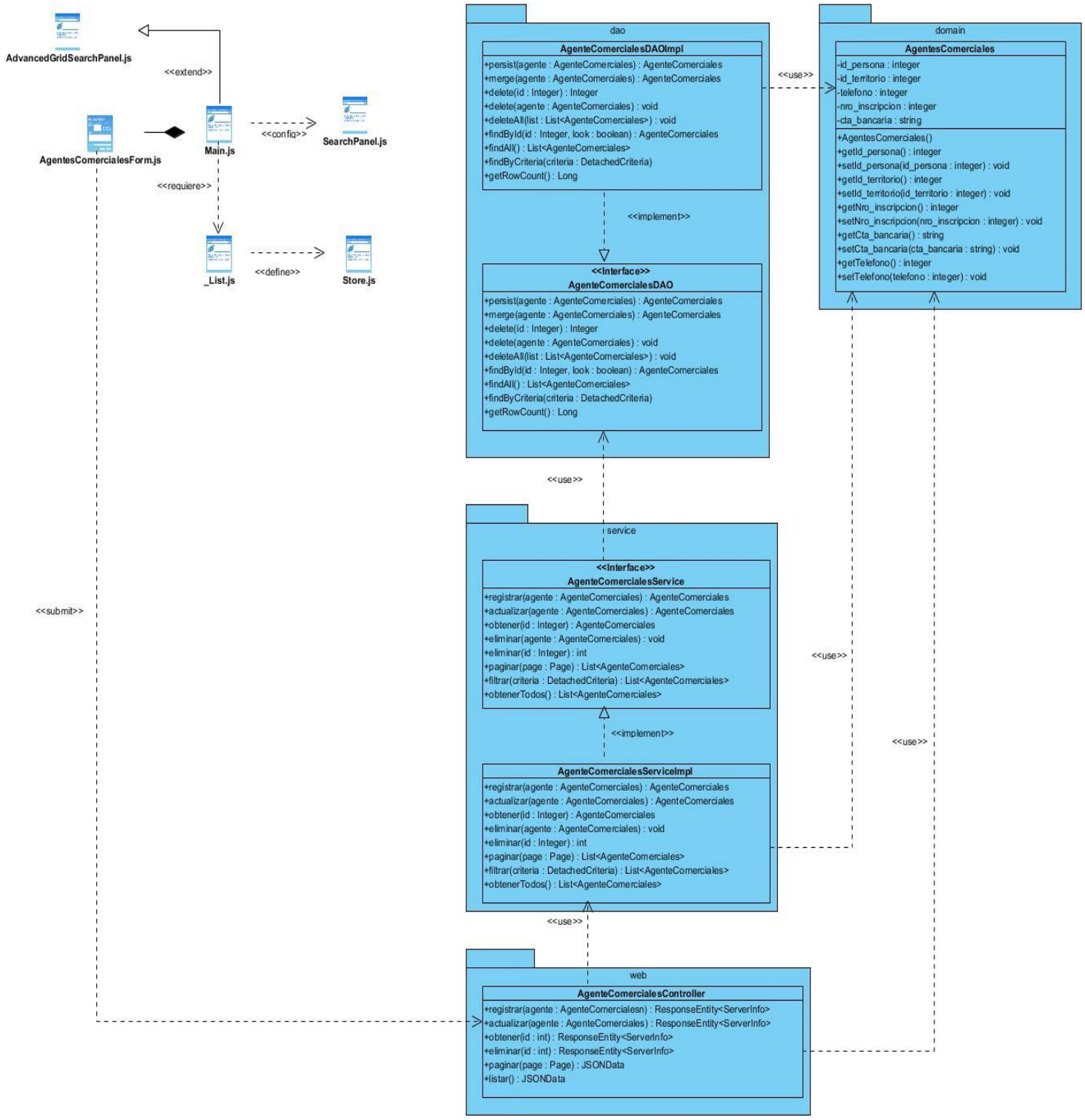
Sommerville, Ian. Software Engineering, 8th ed. ISBN 0-321-31379-8. (en inglés)

Tamara Rodríguez Sánchez. 2015. Metodología de desarrollo para la Actividad productiva de la UCI.

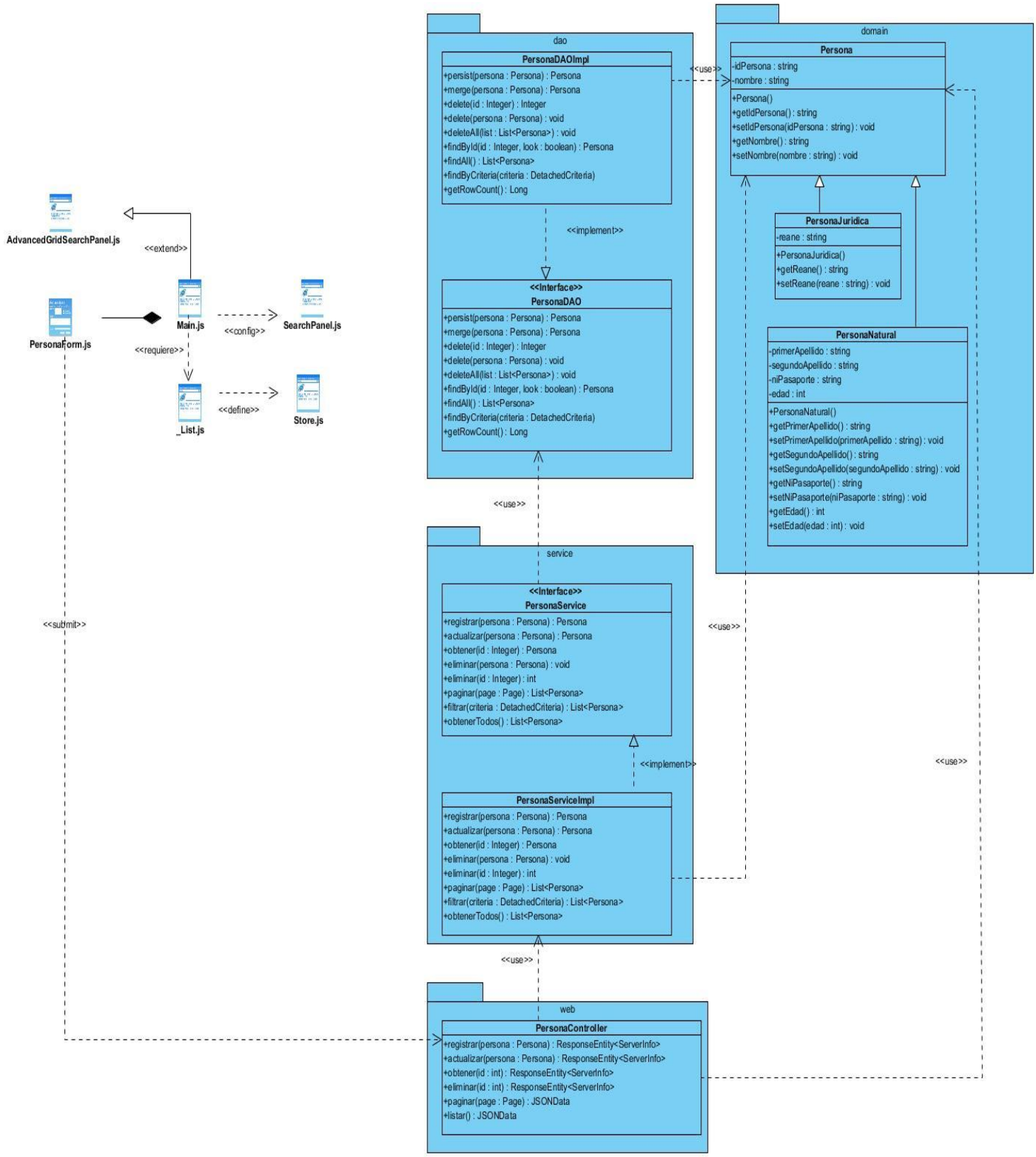
Unidad de Compatibilización Integración y Desarrollo. 2009. Proceso de Desarrollo y Gestión de Proyectos de Software. Ciudad de la Habana: s.n.

WSO2. WSO2 Governance Registry | WSO2. [En línea] [Citado el: 4 de marzo de 2016.] http://wso2.com/products/governance-registry/.

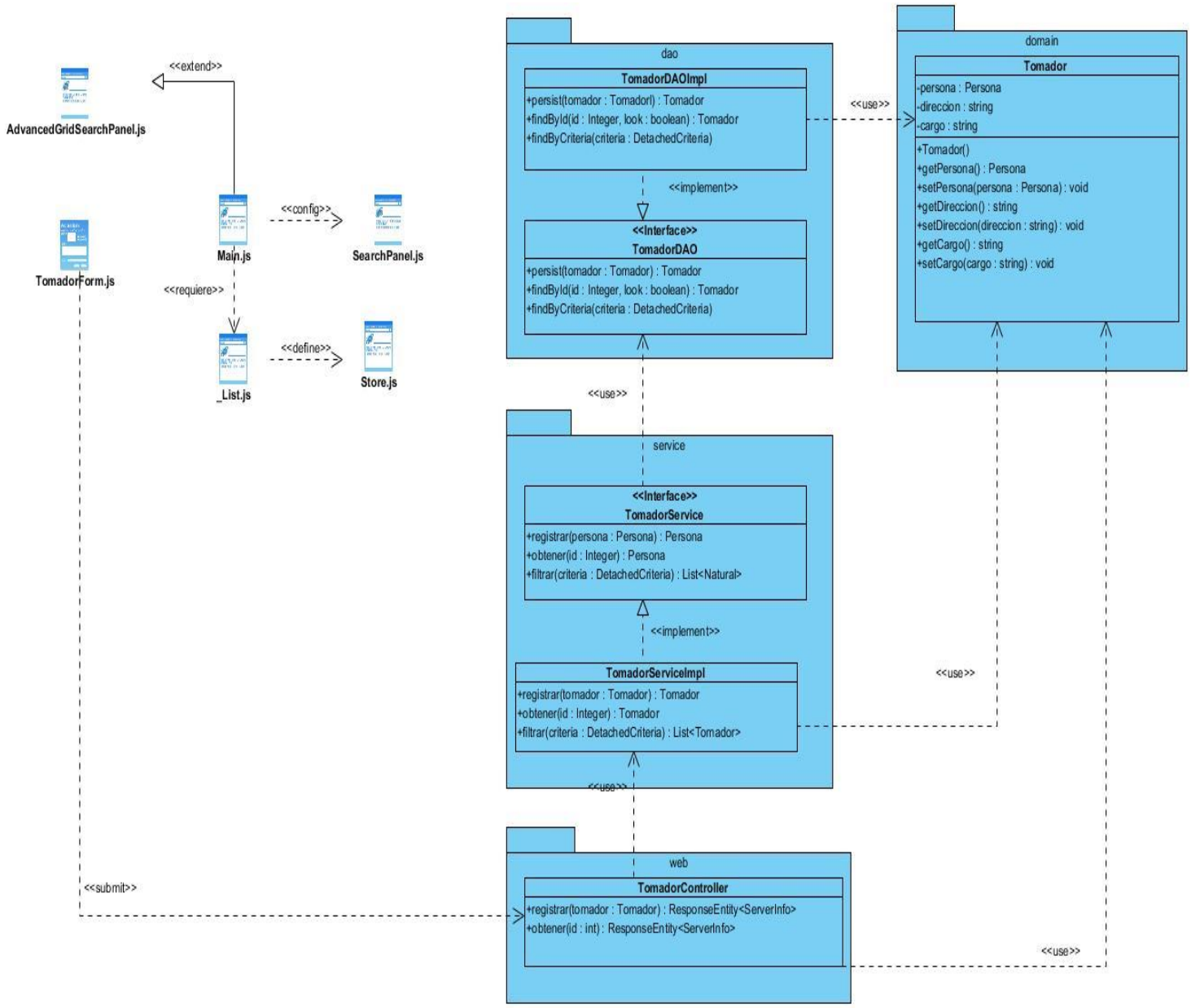
ANEXOS



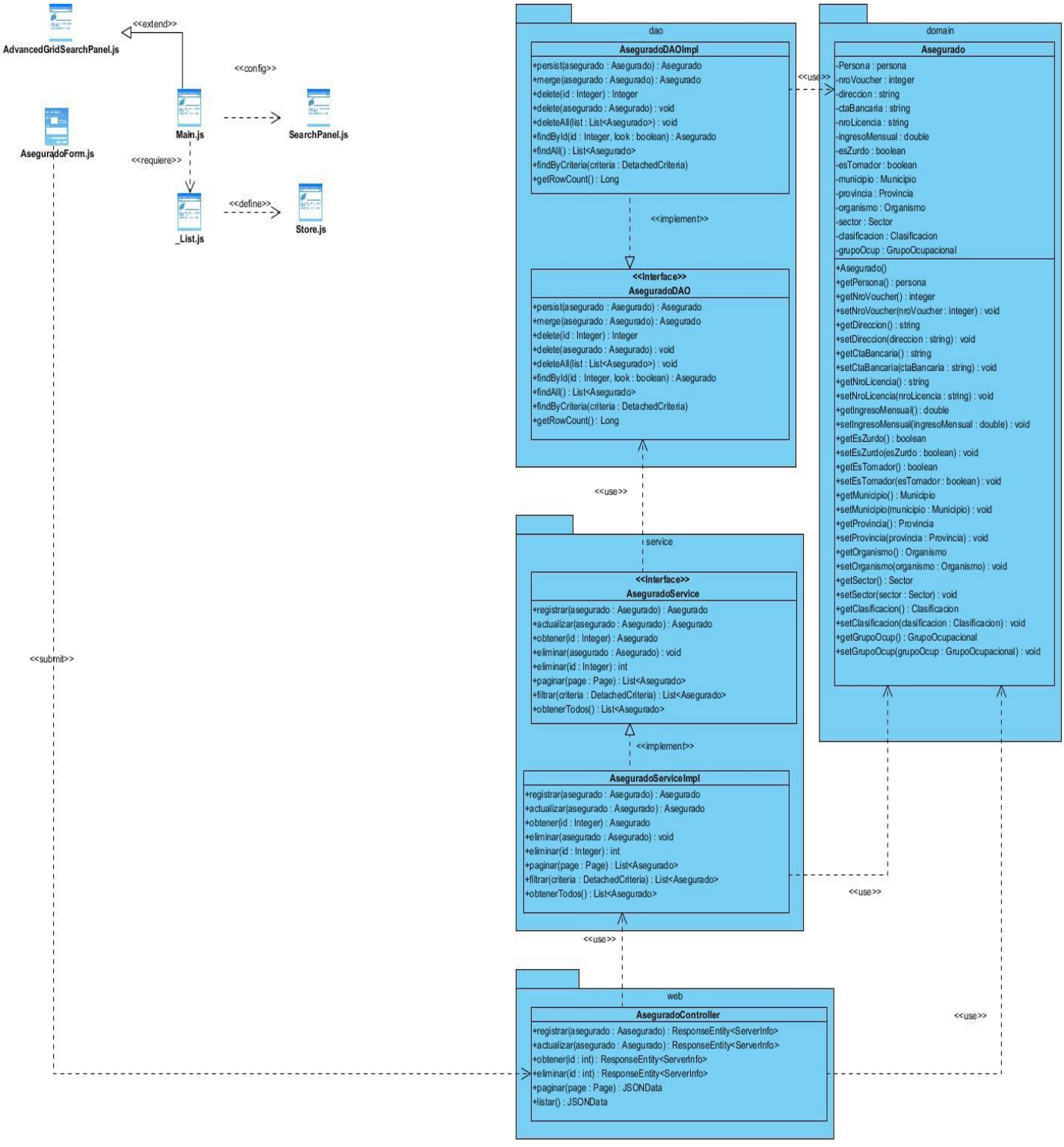
Anexo 1: Diagrama de clases del diseño. Gestionar agentes y comerciales.



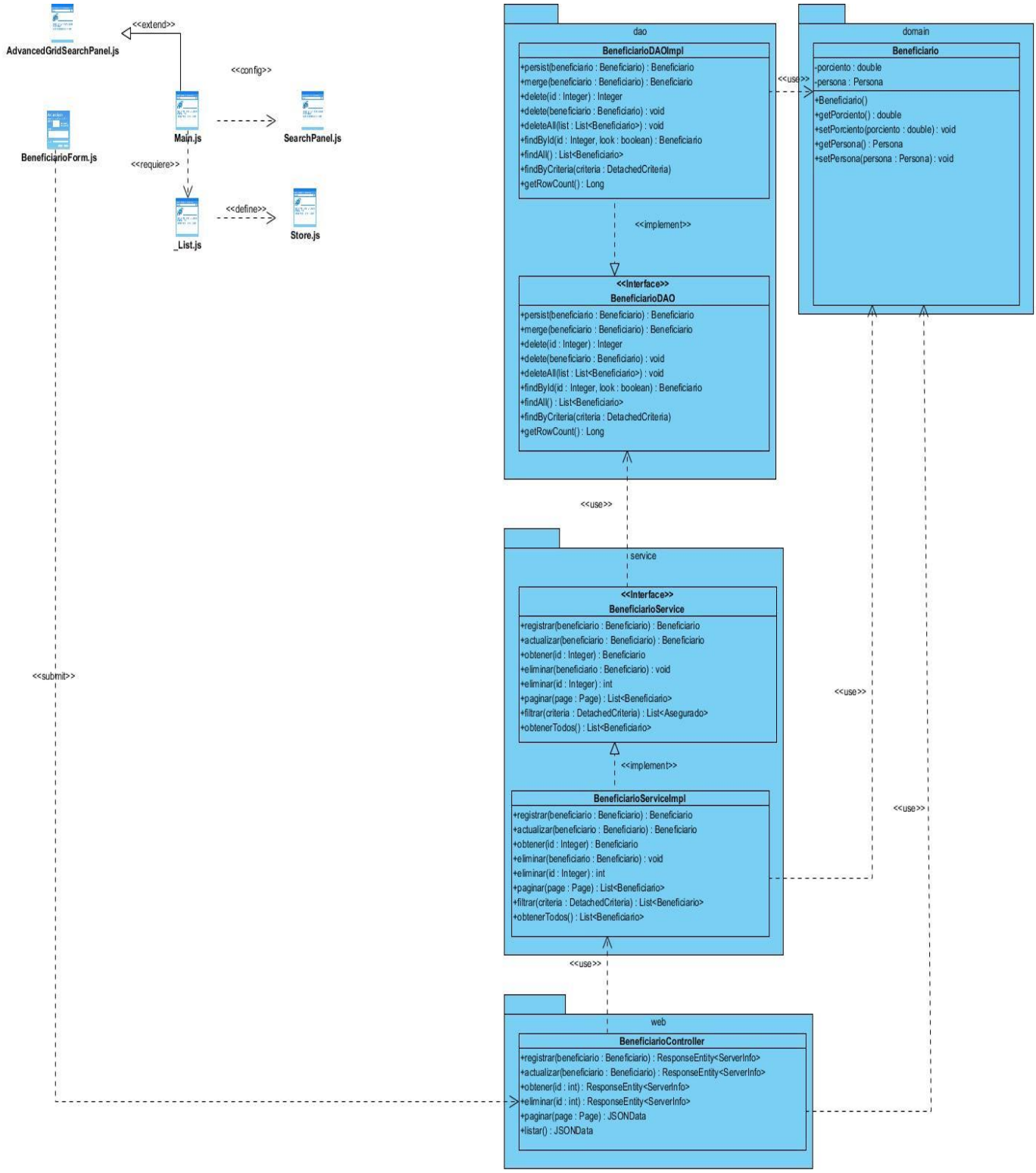
Anexo 2: Diagrama de clases del diseño. Gestionar persona.



Anexo 3: Diagrama de clases del diseño. Gestionar tomador.



Anexo 4: Diagrama de clases del diseño. Gestionar asegurados.



Anexo 5: Diagrama de clases del diseño. Gestionar beneficiarios.

Nro	Clases	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1	Ramo	2	Baja	Baja	Media
2	Modalidad	2	Baja	Baja	Media
3	FormaContratacion	2	Baja	Baja	Media
4	Provincia	2	Baja	Baja	Media
5	Municipio	2	Baja	Baja	Media
6	Organismo	2	Baja	Baja	Media
7	Sector	2	Baja	Baja	Media
8	Clasificacion	2	Baja	Baja	Media
9	Riesgo	2	Baja	Baja	Media
10	Producto	2	Baja	Baja	Media
11	ObjetoSeguro	2	Baja	Baja	Media
12	GrupoOcupacional	2	Baja	Baja	Media
13	FormaPago	2	Baja	Baja	Media
14	ViaFinanciamiento	2	Baja	Baja	Media
15	Moneda	2	Baja	Baja	Media
16	Pais	2	Baja	Baja	Media
17	TipoBien	2	Baja	Baja	Media
18	Poliza	18	Alta	Alta	Alta
19	PolizaSTV	6	Alta	Media	Alta
20	PolizaEnsayoClinico	6	Alta	Baja	Alta
21	PolizaViajeCuba	8	Alta	Baja	Alta
22	PolizaViajeExterior	6	Alta	Baja	Alta
23	Hospital	2	Baja	Baja	Baja
24	Agente	3	Media	Media	Baja
25	Asegurado	28	Media	Media	Media
26	PersonaJuridica	2	Baja	Baja	Media
27	PersonaNatural	4	Baja	Baja	Media
28	Colaborador	2	Baja	Baja	Baja
29	Persona	2	Alta	Baja	Alta
30	Beneficiario	4	Baja	Baja	Media
31	Comercial	2	Baja	Baja	Media
32	Tomador	6	Baja	Baja	Media

Anexo 6: Tabla de resultados de la métrica NCM.

Nro	Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
1	Ramo	1	Bajo	Baja	Alta	Baja
2	Modalidad	1	Bajo	Baja	Alta	Baja
3	FormaContratacion	1	Bajo	Baja	Alta	Baja
4	Provincia	1	Bajo	Baja	Alta	Baja
5	Municipio	1	Bajo	Baja	Alta	Baja
6	Organismo	1	Bajo	Baja	Alta	Baja
7	Sector	1	Bajo	Baja	Alta	Baja
8	Clasificacion	1	Bajo	Baja	Alta	Baja
9	Riesgo	1	Bajo	Baja	Alta	Baja
10	Producto	1	Bajo	Baja	Alta	Baja
11	ObjetoSeguro	1	Bajo	Baja	Alta	Baja
12	GrupoOcupacional	1	Bajo	Baja	Alta	Baja
13	FormaPago	1	Bajo	Baja	Alta	Baja

14	ViaFinanciamiento	1	Bajo	Baja	Alta	Baja
15	Moneda	1	Bajo	Baja	Alta	Baja
16	Pais	2	Medio	Media	Media	Media
17	TipoBien	1	Bajo	Baja	Alta	Baja
18	Poliza	8	Alto	Alta	Baja	Alta
19	PolizaSTV	0	Ninguno	Baja	Alta	Baja
20	PolizaEnsayoClinico	0	Ninguno	Baja	Alta	Baja
21	PolizaViajeCuba	0	Ninguno	Baja	Alta	Baja
22	PolizaViajeExterior	0	Ninguno	Baja	Alta	Baja
23	Hospital	1	Bajo	Baja	Alta	Baja
24	Agente	1	Bajo	Baja	Alta	Baja
25	Asegurado	2	Medio	Media	Media	Media
26	PersonaJuridica	1	Bajo	Baja	Alta	Baja
27	PersonaNatural	1	Bajo	Baja	Alta	Baja
28	Colaborador	1	Bajo	Baja	Alta	Baja
29	Persona	7	Alto	Alta	Baja	Alta
30	Beneficiario	2	Medio	Media	Media	Media
31	Comercial	1	Bajo	Baja	Alta	Baja
32	Tomador	2	Medio	Media	Media	Media

Anexo 7: Tabla de resultados de la métrica CBO.