Universidad de las Ciencias Informáticas

Facultad 3

Módulo de Selectividad para el subsistema Control de Personas de GINA

Trabajo de diploma para optar por el título deIngeniero en Ciencias Informáticas

Autores:
Naisel Rafael Armenteros Piedra
Lianet Camejo Veliz

Tutores:
Ing. Fernando Bermúdez Rodríguez
Ing. Yoandry Freire Pérez
Ing. Nayilet Martín Soler

La Habana, Cuba
Octubre, 2020

Declaración de autoría

Declaramos por este medio que nosotros Lianet Camejo Veliz, con carné de identidad 97092418854 y Naisel Rafael Armenteros Piedra, con carné de identidad 96071307861somos losautores principales de este trabajo titulado “Módulo de Selectividad para el subsistema Control de Personas de GINA” y autorizamos a la facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2020.

_____ _____

Firma de la autoraFirma de autor

Lianet Camejo Veliz Naisel Rafael Armenteros Piedra

_____ _____

Firma del tutor Firma del tutor

Ing. Fernando Bermúdez RodríguezIng. Yoandry Freire Pérez

_____

Firma de la tutora

Ing. Nayilet Martín Soler

Agradecimientos

Lianet Camejo Veliz

Quiero agradecerles a mis padres por traerme al mundo y convertirme en la persona que soy, por apoyarme en todas mis decisiones, y estar siempre para mi cuando lo he necesitado.

A mis tios Tino, Alfonso y tia Celina por estar siempre a mi lado y apoyarme, por convertirse en otros padres para mi gracias a ellos he podido realizar este sueño.

A mi hermana Leydis y mi cuñado por aconsejarme, por preocuparse y estar ahi para mi.

A mis tutores principalmente a Yoandry y Nayilet por no rendirse y por estar siempre para cuando los he necesitado son los mejores del mundo.

A mis dos amigos inseparables Marquito y Yusnavi, por aguantar todos los diasmis pesadeces, siempre estuvieron a mi lado no importa si estuviese de mal humor, triste o alegre, por ayudarme en todo.

A mi amiga Anita por aconsejarme siempre y estar ahi a cualquier hora para escuchar todos mis problemas y locuras.

En general toda mi familia que de una forma u otra me apoyaron, a mis compañeros de aula, los de 1er año y los de 2do en adelante, a todas las personas que han formado parte de mi vida muchas gracias.

Naisel Rafael Armenteros Piedra

Primeramente, quiero comenzar agradeciendo a mi familia en general, por estar pendiente de mi y de mis estudios desde pequeño.

En especial a mis padres, por todo el amor, el cuidado y apoyo que me han dado durante tantos años.

Agradecimientos

También a mi hermano, que no me da consejos ni nada, pero está ahí siempre que lo necesito.

A mis tutores por ser los mejores del mundo, en especial a mi tutor Yoandry por estar, ayudarme siempre cuando he necesitado de su ayuda a la hora que sea.

A mi compañera de tesis Lianet, si no fuese por ella y su insistencia no hubiésemos avanzados en la tesis, porque ella sabe que soy un poco finalista. De verdad que gracias.

A mi pequeña familia que viene desde la infancia, en especial a Dailín, Junior, Nelson, Diego, Raimer, por ser incondicionales conmigo y tenerlos a mi lado en los peores momentos.

A todas mis amistades de la uci que vienen desde 1ro y a los que he conocido con el tiempo, gracias por las risas, las fiestas, las noches de estudio, por dejarme formar parte de sus vidas.

A la pequeña familia que me llevo de la uci Chris, Nailee, Adrián, Cristhian, Loandry y Lianet, que fueron un gran apoyo para mi este último año.

Por último, a mis compañeros de apartamento, al Consejo: Asley, Leonel, Adrian.

Dedicatoria

Lianet Camejo Veliz

Este trabajo va dedicado a mis padres Lidia y Anibal, mis tios Alfonso y Tino, mi hermana Leydis, mi abuela Lulú, mis sobrinos Rubencito y Lesly y a mis amistades.

Naisel Rafael Armenteros Piedra

Este trabajo va dedicado a mis padres Mercedes y Rafael, a mi hermano Yasmany y a mis familiares y amigos.

Resumen

El sistema de Gestión Integral de Aduanas es un software desarrollado en la Universidad de la Ciencias Informáticas, para la Aduana General de la República. Su propósito es informatizar los procesos que se llevan a cabo en todas las áreas de esta institución, entre ellas el área de Lucha Contra el Fraude, permitiéndole así contar con toda la información de los controles realizados a las personas al cruzar la frontera del país. El sistema cuenta con 9 subsistemas que viabilizan el trabajo de los especialistas de esta dirección. El subsistema Control de Personas, encargado de gestionar la información de las personas que ingresan o salen del país,no cuenta con todas las funcionalidades requeridas, por lo que muchos procesos se realizan de forma manual. La presente investigación tiene como objetivo desarrollar el módulo Selectividad, el cual será integrado al subsistema anteriormente mencionado.Brindándole la posibilidad a los analistas de Aduana de detectaraquellos pasajeros susceptibles a cometer o no alguna infracción, a partir de perfiles de riesgos creados por los oficiales del área de Enfrenamiento con la información adelantada de los pasajeros de un vuelo. El desarrollo de la solución está guiado por el uso de la metodología de desarrollo de software Proceso Unificado Ágil en su variación para la Universidad de las Ciencias Informáticas y la utilización de tecnologías de código abierto.

Palabras clave: perfil,riesgo, selectividad,viabilizan.

Índice de contenidos

Índice de figuras

Índice de figuras

Índice de tablas

Introducción

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) han alcanzado un alto grado de desarrollo, lo que permite la mejora de las comunicaciones humanas, siendo estas un factor decisivo en el origen de los problemas sociales. La implementación de dichas tecnologías en la sociedad contribuye a elevar la calidad de la información en función de lograr mejor eficiencia, así como la mejora en la velocidad de las comunicaciones y el aumento del suministro de la información de valor para el conocimiento de las perspectivasen todas las esferas de la sociedad. Cuando se emplean las TIC, el tiempo en el almacenamiento, consulta, trasmisión y procesamiento de información, así como los errores humanos disminuyen considerablemente, superando a los procedimientos tradicionales donde los soportes de información están sometidos a un tiempo de deterioro mucho más rápido [1].

Por lo anteriormente expuesto se recomienda recurrir al uso de las TIC en cada uno de los procesos que se realizan en las entidades que son de relevancia para el desarrollo de un país.Cuba se ha sumado a este proceso de informatización de la sociedad, donde en colaboración con laUniversidad de las Ciencias Informáticas (UCI) se desarrollan proyectos de informatización en varias entidades,ejemplo de ello es la Aduana General de la República(AGR).La misma es un órgano de la administración del estado que tiene entre sus funciones principales la de participar en la elaboración y hacer cumplirlas disposiciones relativas a regulaciones, restricciones y requisitos especiales aplicables a las importaciones y exportaciones por razones de protección económica, ambiental, sanitaria, patrimonial y de orden público. La AGR fue creada con el objetivo de garantizar la seguridad nacional. Con este propósito se desarrolla en ella, el área de Lucha Contra el Fraude (LCF), encargada de enfrentar las acciones terroristas, de narcotráfico y las que ponen en riesgo el patrimonio cultural y natural del país, lo cual queda reglamentado en el Decreto Ley No. 162 de Aduanas, en su artículo 16, inciso c).Establece entre las atribuciones y funciones especiales de la Aduana, la de prevenir, detectar y enfrentar el fraude comercial, el contrabando y otras infracciones aduaneras en el desarrollo del tráfico comercial de mercancías, viajeros y envíos por vía aérea, marítima y postal [2].

En los últimos años, la Aduana al igual que el resto de las entidades del país, ha mostrado un gran interés en informatizar los procesos que en ella se desarrollan. Por tanto, se implantaron varios sistemas informáticos que automatizaban algunos de los procesos aduanales, pero con el paso de los años fueron quedando obsoletos. Por estos motivos se desarrolló un nuevo sistema encargado de automatizar los

procesos del área de LCF permitiendo a la AGR contar con toda la información sobre los controles realizados a las personas al cruzar la frontera del país.

En colaboración con el Departamento de Soluciones para la Aduana del Centro de Informatización y la Gestión de Entidades (CEIGE) perteneciente a la UCI y el Centro de Automatización y Dirección de la Información (CADI) se desarrolla el sistema de Gestión Integral Aduanera (GINA). El cual está compuesto por 9 subsistemas que abarcan los procesos que gestiona la AGR. Uno de estos subsistemas es el subsistema Control de Personas que gestiona la información de todas las personas que entran o salen del país, para garantizar la protección técnica en frontera. Entresus funcionalidades más importantes se encuentra el estudio de la información adelantada de pasajeros (EAPI), donde se realiza un estudio adelantado de los datos personales y de viaje de cada uno de los pasajeros de un vuelo.

Aunque el subsistema es muy útil, no cuenta con una funcionalidad que luego de realizar dicho estudio le permita detectar los posibles infractores que pudieran estar relacionados con el tráfico de drogas, lavado de dinero o terrorismo.Actualmente este proceso de detección se realiza por el analista de la Aduana de forma manual y empírica en un período de tiempo muy corto aproximadamente de 5 a 10 minutos antes del arribo del vuelo, debido al comportamiento cambiante de la información. Por lo que representa un retraso en el proceso debido al cúmulo de información ya que los vuelos que arriban al país, principalmente en el aeropuerto José Martí suelen llegar con una diferencia de 7 a 10 minutos, con aproximadamente un promedio de 100 a 150 pasajeros.

Para el análisis de la información adelantada de pasajeros (API) se hace un estudio histórico de la información registrada, si el pasajero ha entrado o no al país en otras ocasiones y cuántas veces, si tiene o no infracciones y cuáles son, si tiene marcajes y si tiene alguna categoría de interés aduanal. Es posible identificar infractores en un vuelo debido a la experiencia acumulada, dada la sucesión histórica de los hechos delictivos,donde se ha evidenciado un comportamiento distintivo entre ellos, posibilitando identificar intuitivamente varias características entre los pasajeros y luego realizar un marcaje por el EAPI, persona de interés aduanal-persona de interés aduanal control (PIA-PIA control) o requerimiento puntual. Este proceso representa una gran cantidad de información a procesar para la realización del marcaje. Todo esto trae consigo que disminuya el potencial de efectividad del analista, aumente el riego de omisión de posibles infractores de un vuelo y dificulte la obtención de algún dato estadístico sobre los posibles infractores.

A partir de la problemática antes descrita surge la necesidad de resolver el siguiente **problema de investigación:**¿cómo contribuir con la detección de pasajeros susceptibles a cometer infracciones en la Aduana General de la República?

Definiéndose como **objeto de estudio:**proceso de selección de pasajeros.

Identificándose como **campo de acción:**información adelantada de pasajeros en la Aduana General de la República.

Definiéndose como **objetivo general:**desarrollar el módulo de Selectividad para el subsistema Control de Personas, para contribuir con la detección de pasajeros susceptibles a cometer infracciones en la Aduana General de la República.

A partir de este objetivo general se desglosan los siguientes **objetivos específicos:**

- Realizar un estudio acerca de los conceptos fundamentales del proceso de negocio para la definición de los mismos.
- Realizar un estudio y familiarización de las tecnologías, lenguajes y herramientas establecidas para la construcción del proceso de desarrollo de software.
- Diseñar el módulo propuesto a partir del proceso de desarrollo de software seleccionado.
- Implementar el módulo propuesto a partir de los requisitos identificados.
- Validar el módulo propuesto a partir de la ejecución de pruebas de software.

Teniendo como **hipótesis:**si se desarrolla el módulo de Selectividad para el subsistema Control de Personas, se contribuirá con la detección de pasajeros susceptibles a cometer infracciones en la Aduana General de la República.

Luego de haberse planteado el marco teórico referente a las necesidades de la AGR se debe realizar un estudio e investigación para dar solución al problema de investigación planteado, haciendo uso de los siguientes métodos científicos de investigación:

Métodos teóricos

Histórico-Lógico:permite constatar teóricamente cómo ha evolucionado un fenómeno en un período de tiempo dado. Se utiliza para estudiar las formas de solución de problemas similares a la detección de pasajeros susceptibles a cometer infracciones en la Aduana.

Analítico-Sintético:se emplea para el análisis de teorías y documentos, extrayéndose cada uno de los aspectos esenciales de las herramientas y la literatura seleccionada para una correcta elaboración del módulo propuesto.

Métodos empíricos

Entrevista:permite establecer un intercambio con el cliente para comprender sus necesidades, así como el funcionamiento de los procesos de negocio de la organización, con el objetivo de definir el problema existente, a partir de la situación problémica y capturar los requisitos correspondientes al módulo propuesto. Para la visualización de la misma se debe verel anexo 1.

Estructura del documento

El presente trabajo de diploma consta de 3 capítulos que abarcan todo el proceso de desarrollo del módulo propuesto los cuales están estructurados de la siguiente forma:

Capítulo 1: Fundamentación teórica.

En este capítulo se definen los fundamentos teóricos de la investigación. Se proponen los conceptos fundamentales relacionados con el negocio definidos por el cliente, luego de haber realizado las técnicas de obtención de requisitos. Se describen sistemas aduaneros que fueron implantados anteriormente en la AGR.Se realiza una descripción de las herramientas, tecnologías, lenguajes de programación, metodología de desarrollo de software y marco de trabajo que se utilizan en la implementación del módulo propuesto, así como el diseño arquitectónico empleado. Además, se describen las técnicas de captura y validación de requisitos que se utilizarán.

Capítulo 2: Análisis, diseño e implementación del módulo propuesto.

En este capítulo se definen los principales elementos a tener en cuenta para la implementación del módulo propuesto.Se exponeel modelo conceptual, el modeladodel proceso de negocio para una mejor comprensión del mismo, se lleva a cabo el proceso de captura de los requisitos funcionales y no funcionales aplicables para la implementación, así como la descripción de requisitos por proceso. Se realiza el modelo de datos y el empleo de patrones de diseño. Se establecen los estándares de codificación para un mejor desarrollo del módulo propuesto, así como la implementación. Se describe el módulo implementado, con sus funcionalidades e interfaces.

Capítulo 3: Validación del módulo propuesto.

En este capítulo se ponen en práctica las técnicas de validación de requisitos descritas en el capítulo 1, se evalúael correcto funcionamiento de cada una de las funcionalidades del módulo, haciendo uso de las pruebas de softwaredefinidas para realizar la validación del módulo, como pruebas unitarias, pruebas de aceptación y pruebas de integración. Se aplicarán cada uno de estos niveles de prueba según el tipo prueba y sus respectivos métodos y técnicas a realizar.

Capítulo 1: Fundamentación teórica

En el presente capítulo se definen los fundamentos teóricos de la investigación. Se establecen los conceptos que hacen posible el entendimiento del negocio. Se proponen características de sistemas informáticos aduaneros que han sido utilizados y dieron paso a la realización de un sistema más completo como lo es GINA. Se realiza un estudio de las principales herramientas, tecnologías y lenguajes utilizados para la implementación del módulo propuesto, así como la metodología de desarrollo de software que guiará el proceso. Además, se realiza una breve descripción del diseño arquitectónico, las técnicas de captura y validación de requisitos que serán aplicadas posteriormente.

1.1. Conceptos asociados al dominio del problema

Dentro del marco de los procesos que se llevan a cabo en la AGR es muy importante conocer sobre los conceptos aduaneros que se ponen en práctica. El análisis de los mismos permite el entendimiento de los procesos del negocio. Por tanto, se hace necesario conocer el significado de cada uno de ellos, producto de la entrevista realizada (ver anexo 1).

El**estudio de la información adelantada de pasajeros (EAPI)**es el análisis de la información previa sobre los pasajeros de un vuelo.

Selectividades el proceso de agrupar datos de personas al aplicar perfiles de riesgo sobre un EAPI.

Un **perfil de riesgo** es un conjunto de criterios de selección definidos, está compuesto por una fecha de inicio, una descripción, una línea de enfrentamiento, una modalidad y medidas a tomar. La aplicación de los mismos permite la selección de personas que pudieran estar relacionado con alguna infracción.

Los **criterios de selección** se desglosan como la relación dada por un indicador de riesgo, un operador de riesgo y su valor. Estos criterios se basan en los datos personales y de viaje de las personas que arriban en un vuelo y son definidospor la Aduana.

Los **indicadores de riesgo** son criterios de selección específicos definidos por la Aduana, producto de las diferentes características o propiedades asociadas alas personas en un EAPI. Estas características se han evidenciado en los diferentes hechos delictivos que han ocurrido en el país.

Un **operador de riesgo** en la AGR serían operadores de comparación para asociar indicador-valor.

El siguiente mapa conceptual explica la relación de los conceptos asociados para un mejor entendimiento de los procesos del negocio.

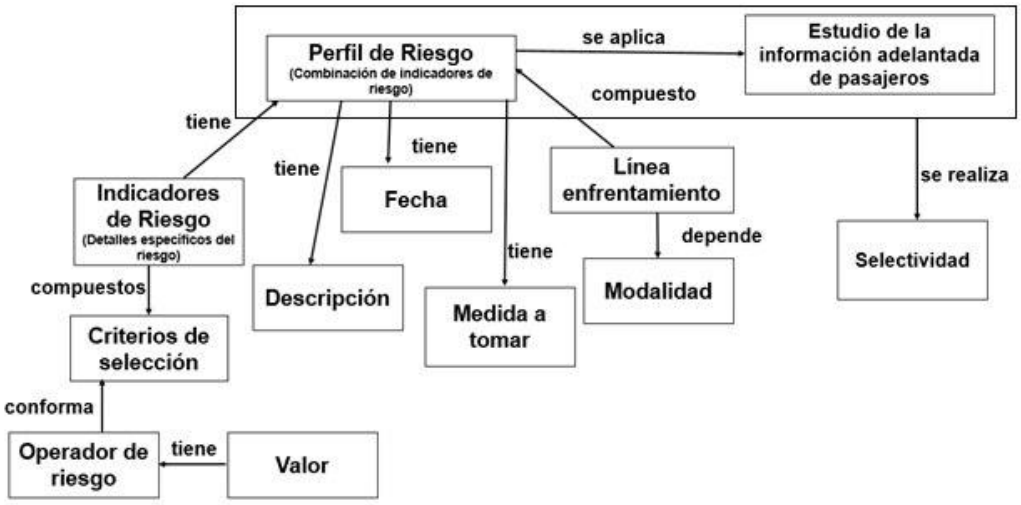

Figura 1. Mapa conceptual

Fuente: Elaboración propia

1.2. Sistemas aduaneros

Como parte del proceso de modernización del servicio de Aduanas a través de la automatización basada en las TIC se comenzó a emplear en la AGR sistemas informáticos que fueron permitiendo agilizar y perfeccionar el control aduanero en algunas áreas de trabajo. Estos sistemas solo realizaban algunas tareas específicas, era necesario contar con un sistema que integrara todas las plataformas de trabajo aduanero en uno y además incluyera la selectividad de personas como unos de los procesos fundamentales para la detección de pasajeros que pudieran estar relacionados con algún hecho delictivo que pusiera en peligro el estado y la soberanía nacional.

Sistemas internacionales

AIRCOP

El Proyecto de Comunicaciones Aeroportuarias(AIRCOP) fue diseñado originalmente en el 2010, es un sistema financiado por la Unión Europea (UE) bajo el Programa de la UE de la Ruta de la Cocaína e implementado por la Oficina de las Naciones Unidas contra la Droga y el Delito (UNODC), en colaboración con INTERPOL y la Organización Mundial de Aduanas (OMA)es una iniciativa anti-tráfico, multi-

agencialque apunta a fortalecer las capacidades de detección, interdicción e investigación de los agentes de seguridad en los aeropuertos participantes. Contribuye en la lucha contra el crimen transnacional organizado y hace frente a los desafíos de un sistema legal fragmentado, promoviendo la cooperación regional y transregional. Su objetivo es crear grupos operativos inter-agenciales para fortalecer las capacidades de los aeropuertos internacionales para detectar e interceptar drogas, otros vienes ilícitos y pasajeros de alto riesgo, incluidos los combatientes terroristas extranjeros en los países de origen, tránsito y destino con el objetivo de general de interrumpir las redes delictivas transnacionales. Actualmente 26 países de África, América Latina y el Caribe participan en el proyecto los cuales han registrado hasta la fecha resultados impactantes a nivel global. La UNODC organizó un entrenamiento especializado en la detección de pasajeros con perfiles de alto riesgo, en el marco del proyecto AIRCOP, realizado en la Ciudad de Panamá, Panamá. Este entrenamiento se enfocó en el desarrollo de las diferentes técnicas de perfilamiento de pasajeros, indicadores de alerta, captación de la información, entrevistas, análisis de riesgo, procedimientos de detección de la verdad y estrategias y mecanismos para luchar contra el crimen organizado y el terrorismo internacional [3].Aunque este sistema cumple con sus funciones aún está implementándose, además es un software privado, no es código abierto, el acceso a su funcionamiento interno incurriría en costos para el país. No obstante, el estudio detallado de este sistema sirvió de base para el desarrollo de la propuesta de solución.

SIDUNEA

Es un sistema aduanero automatizado desarrollado por la Conferencia de las Naciones Unidas sobre el Comercio y el Desarrollo (UNCTAD),es la herramienta informática para el control y administración de la gestión aduanera, actualmente esusada con éxito en más de 80 países. En la actualidad la UNCTAD ofrece tres versiones del sistema,las cuales son [4]:

- SIDUNEA World
- SIDUNEA++
- SIDUNEA v2

Este sistema cubre todo el proceso de declaración y procesamiento, incluida lacarga y el tránsito. Utiliza herramientas sofisticadas, desde la clásica selección delprocedimiento de examen y la asignación de las mercaderías declaradas a un “canal” decontrol (verde para la liberación de las mercaderías sin examen; amarillo para el controlde la documentación antes de la liberación de las mercaderías; rojo para el examen

físico de las mercaderías antes de la liberación; azul, para indicar que las mercaderíasserán liberadas, pero quedarán sujetas a un control de auditoría post-despacho por partede la aduana) hasta el uso de multimedios, imágenes escaneadas y aparatos decomunicación inalámbrica. Los controles aduaneros se pueden llevar a cabo ensituaciones que no eran posibles anteriormente, por ejemplo, para detener carga entránsito y verificar que los documentos en papel presentados corresponden a lodeclarado al momento de la partida o para realizar controles sorpresivos de loscontenidos de un contenedor y el estatus de las mercaderías (despachadas, en tránsito,). El sistema permite la evaluación periódica del proceso de administración de riesgoa fin de medir la efectividad de los criterios de selectividad de mercancíasy para cambiar, extenderoeliminar los parámetros de administración de riesgo, según fuere necesario [4]. Este sistema, aunque es muy efectivo, no se adecuaba 100% a las leyes y regulaciones cubanas, además el país pagaba grandes sumas de dinero por conceptos de licencia de software.

Sistemas nacionales

SAPIA

Sistema Automatizado de Interés Aduanal fue utilizado para gestionarlas personas naturales que cometen infracciones y las que viajan con frecuencia al país, por lo que quedaba sin analizar las personas extranjeras que entraban y salían por las fronteras.

SACOM

Sistema Automatizado de Control Mercantil, como su nombre lo indica era el encargado de monitorizar las actividades de control mercantil en el área LCF. Era un sistema solo especializado en esta funcionalidad del área LCF.

Conclusiones del estudio de los sistemas

Según la entrevista realizada al cliente estos sistemas no cumplían con todos los requisitos necesarios para lograr un mejor resultado por parte de los trabajadores de la Aduana ya que estos necesitaban nuevas y mayores prestaciones. Fueron quedando obsoletos debido al auge de las tecnologías en el mundo, y presentando desventajas a la hora de realizar el mantenimiento de los mismos, intercambiar información o tratar de evolucionar hacia una nueva versión; ya sea por el cambio en las normativas de la Aduana o por el desarrollo acelerado que ha tenido el mundo de la informática en la actualidad. Además de la necesidad de contar con una funcionalidad que le permitiera detectar las personas con más probabilidades a cometer alguna infracción.

1.3. Metodología de desarrollo de software

Una metodología de desarrollo de software, consiste principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo de software, actualmente existen mucha variedad en metodologías de programación, donde todas están basadas en ciertos enfoques generalistas que se crearon hace muchos años. Su objetivo es aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo [5].

No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Están divididas en dos grupos, las ágiles y las tradicionales. Las metodologías ágiles son aquellas que se encargan de adaptar el proceso del trabajo a las circunstancias y contexto en el que se encuentra, para que, si ocurre algún inconveniente o cambio inesperado en el panorama, los procedimientos puedan adaptarse con facilidad y de manera inmediata, y así el proyecto no se vea afectado negativamente de ninguna manera [5].

Es por ello que, en la presente investigación se empleará la metodología Proceso Unificado Ágil (por sus siglas en inglés AUP), esencialmente la variación realizada en la UCI para guiar los procesos productivos, adaptada al ciclo de vida de los proyectos, dado que el módulo formará parte del proyecto GINA donde existe una documentación precedente del uso de la misma. A continuación, se plantea una descripción de los principales elementos de dicha metodología.

1.3.1. Metodología AUP-UCI

Debido al ciclo de vida definido para la actividad productiva de la UCI se decide hacer una variación de la metodología AUP apoyado en el Modelo CMMI-DEV v1.3[1]. El cual 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[6].

1.3.2. Descripción de las fases de la metodología

[1] CMMI-DEV v1.3: Integración del modelo de madurez de capacidades para el desarrollo (por sus siglas en inglés *CapabilityMaturityModelIntegrationforDevelopment*). Este modelo proporciona un conjunto completo e integrado de guías para desarrollar productos y servicios.

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma. Se unifican las restantes 3 fases de AUP en una sola, a la que se le llamará Ejecución y se agrega una fase de Cierre [6]. Para una mayor comprensión se muestra la siguiente tabla.

Tabla 1. Descripción de las fases de la metodología AUP-UCI

Fases AUP	**Fases variación AUP-UCI**	**Objetivos de las fases (AUP-UCI)**
Inicio	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 [6].
Elaboración Construcción Transición	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. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software [6].
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto [6].

Fuente: Metodología de desarrollo para la actividad productiva UCI

1.3.3. Escenarios de la metodología

El modelado de negocio propone tres variantes a utilizar en los proyectos (Caso de Uso del Negocio (CUN), Diagrama de Proceso del Negocio (DPN) y Modelo Conceptual (MC)). Existen tres formas de encapsular los requisitos (Caso de Uso del Sistema (CUS), Historias de Usuarios (HU) y Diagrama de Requisitos por Proceso (DRP)), a partir de ellos surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma [6]:

Escenario No 1:

Proyectos que modelen el negocio con CUN sólo pueden modelar el sistema con CUS.

Escenario No 2:

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

Escenario No 3:

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

Escenario No 4:

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

Esencialmente se estará utilizando el escenario número 3 de dicha metodología para regir el desarrollo del módulo propuesto ya que este escenario, se 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 [6].

1.4. Herramientas, tecnologías y lenguajes

Para la implementación del módulo propuesto se decide utilizar las herramientas y tecnologías que establece el proyecto GINA, pues el módulo forma parte de unos de sus subsistemas. A continuación, se presenta una breve descripción de las características que definen a cada una de ellas dándole respuesta al porqué de su utilización.

Herramientas de Ingeniería del Software Asistidas por Computadoras

Las herramientas de ingeniería del software asistidas por computadoras (por sus siglas en inglés CASE*Computer Aided Software Engineering*)son un conjunto de aplicaciones informáticas, usadas para automatizar actividades del ciclo de vida de desarrollo de sistemas, son usadas en algunas de las fases de desarrollo de sistemas de información, incluyendo análisis, diseño y programación. Su objetivo fundamental es proveer un lenguaje para describir el sistema general que sea lo suficientemente explícito para generar todos los programas necesarios. Las CASE aceleran el desarrollo del proyecto con tal de producir los resultados deseados y ayuda a encontrar imperfecciones antes de proseguir con la siguiente etapa del desarrollo de software [7]. Según el autor Ian Sommerville, la define como “el software que se utiliza para ayudar a las actividades del proceso del software como la ingeniería de requisitos, el diseño, el desarrollo de programas y las pruebas. Las herramientas CASE incluyen editores de diseño, diccionarios de datos, compiladores, depuradores, herramientas de construcción de sistemas”.

La herramienta CASE a utilizar en la presente investigación es el Visual Paradigm, en el siguiente epígrafe se expondrán algunas características de dicha herramienta.

Herramienta CASE Visual Paradigm 8

Visual Paradigm es una herramienta CASE de modelado visual que facilita la construcción de artefactos en un proceso de desarrollo de software mediante el lenguaje de modelado unificado (UML). Soporta una amplia gestión de casos de uso y diseño de base de datos relacionales y proporciona medidas más eficaces en el análisis y diseño de sistemas [8].

Algunas de las funcionalidades que brinda la herramienta son [8]:

- La generación de código y de base de datos a partir de los diagramas UML realizados.
- La realización de ingeniería inversa.
- Generación automática de informes en formato PDF[2], Word[3] o HTML[4].
- Generación de máquinas de estados, integración con Visio y Rational Rose".

Tomando en cuenta la existencia de una Licencia UCI para el uso de esta herramienta y las características de la misma, para el desarrollo del presente trabajo se utiliza en su versión 8, unida al lenguaje de modelado unificado (por sus siglas en inglés UML*Unified Modeling Language*) el cuál se explica a continuación.

Lenguaje de modelado unificado UML 2.0

Es un lenguaje de modelado estandarizado que consiste en un conjunto integrado de diagramas, desarrollado para ayudar a los desarrolladores de sistemas y software a especificar, visualizar, construir y documentar los artefactos de los sistemas de software, así como para el modelado comercial y otros sistemas de software. El UML representa una colección de mejores prácticas de ingeniería que han demostrado ser exitosas en el modelado de sistemas grandes y complejos. Es una parte muy importante del desarrollo de software orientado a objetos y el proceso de desarrollo de software, utilizando principalmente notaciones gráficas para expresar el diseño de proyectos de software. El uso de UML ayuda a los equipos de proyecto a comunicarse, explorar diseños potenciales y validar el diseño arquitectónico del software [9].

El lenguaje de modelado unificado está compuesto por tres clases de bloques de construcción [10]:

- Elementos: los elementos son abstracciones de cosas reales o ficticias (objetos, acciones).
- Relaciones: relacionan los elementos entre sí.
- Diagramas: son colecciones de elementos con sus relaciones.

Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento

[2] PDF: Formato de documento portátil (por sus siglas en inglés *Portable DocumentFormat*).

[3]WORD: Es un programa de computadora que sirve para crear, modificar, e imprimir documentos escritos. A este tipo de programa se le conoce como Procesadores de texto y son los más comunes de entre todas las aplicaciones de computadoras.

[4]HTML: Lenguaje de marcos de hipertextos(por sus siglas en inglés *Hyper text markup languaje*).

físico y lógico de una máquina. Permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias [11].

Se estará utilizando en la presente investigación Procesador de hipertexto (por sus siglas en inglés PHP *Hypertext Preprocessor)*como lenguaje de programación, a continuación, se explican algunas características principales de dicho lenguaje.

PHP 5.4

PHP es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno [11].

PHP es conocido como un lenguaje público y con numerosas características [12]:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a informaciónalmacenada en una base de datos.
- Tiene capacidad de conexión con la mayoría de los motores de bases de datos que se utilizan en la actualidad, destaca su conectividad con MySQL[5] y PostgreSQL[6].
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables, aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.

Librería Ext js de Java Script 3.0

[5]MySQL: Es un sistema de gestión de base de datos relacional de código abierto, basado en lenguaje de consulta estructurado (SQL por sus siglas en inglés *Structured Query Language*).

[6] PostgreSQL: Es un servidor de base de datos objeto relacional libre ya que incluye características de la orientación de objetos.

Ext JS es un marco de JavaScript para crear aplicaciones web y móviles multiplataforma intensivas en datos para cualquier dispositivo moderno. Ext JS incluye más de 115 componentes de interfaz de usuario de alto rendimiento pre-integrados y probados [13]. Se estará utilizando en la presente investigación esta librería en su versión 3.0.

Entorno de desarrollo integrado

Un entorno de desarrollo integrado, llamado también IDE (en inglés de *Integrated Development Environment*),corresponde a un entorno de programación que permite desarrollar de forma integrada el desarrollo y empaquetado de códigos por medio de una aplicación.Lo cual permitirá editar el código fuente de las aplicaciones informáticas, compilar o traducir a un lenguaje legiblepara los distintos tipos de máquinas, depurar, probar o eliminar errores en el desarrollo de un programa, el cual nos permitirá crear y diseñar interfaces entre la aplicación y el usuario informático. Está compuesto por un conjunto de herramientas de programación que puede dedicarse exclusivamente a un solo lenguaje de programación o bien puede utilizarse para varios [14].

Existen varios IDE, pero precisamente el que se estará utilizando en el presente trabajo es PhpStorm.

PhpStorm 2017.2

PhpStorm es un entorno de desarrollo integrado multiplataforma especializado para PHP, ideal para trabajar con Symfony[7], Laravel[8], Drupal[9], WordPress[10], Zend Framework[11] y otros marcos de trabajo. Aprovecha al máximo las últimas tecnologías de *front-end*, tales como HTML 5, CSS[12], Sass[13],con disponibilidad de refactorizaciones, depuración y pruebas de unidad. Soporta integración con el Sistema de control de versiones, compatibilidad con implementación remota, bases de datos/SQL, herramientas de línea de comando, *Docker*, *Composer*, *REST Client* y muchas otras herramientas [15].

Marco de trabajo

Un marco de trabajo es un entorno, plataforma o estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Los marcos de trabajo facilitan el desarrollo de software,

[7]Symfony:Es un marco de trabajo diseñado para desarrollar aplicaciones web.
[8]Laravel: Es un marco de trabajo de código abierto para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7.
[9]Drupal:Es un sistema de gestión de contenidos modular, multipropósito y muy configurables que permite publicar artículos, imágenes, archivos.
[10]WordPress: Es un sistema de gestión de contenidos enfocado a la creación de cualquier página web.
[11]Zend Framework: Es un marco de trabajo de código abierto para desarrollar aplicaciones web. Es una implementación que usa código 100 % orientado a objetos.
[12]CSS: Hojas de estilo en cascada (por sus siglas en inglés *Cascade*)
[13]Sass: Hojas de estilo sintácticamente impresionantes (por sus siglas en inglés *Syntactically Awesome Style Sheets*).

permiten evitar los detalles de bajo nivel, concentrando más esfuerzos y tiempo en identificar los requisitos del software [16].

En la presente investigación se estará utilizando como marco de trabajo Symfony.

Symfony 1.2.8

Symfony es un marco de trabajo desarrollado completamente con PHP para el desarrollo de aplicaciones web, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQLServer de Microsoft [17].

Symfony se diseñó para que se ajustara a los siguientes requisitos [17]:

- Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Sistema Gestor de Bases de Datos

Un sistema de administración de bases de datos (SGBD) es un software de sistema para crear y administrar bases de datos. Permite a los usuarios finales crear, leer, actualizar y eliminar datos en una base de datos, esencialmente sirve como una interfaz entre la base de datos y los usuarios finales o programas de aplicación, asegurando que los datos estén organizados de manera consistente y sean fácilmente accesibles. El SGBD gestiona tres factores importantes: los datos, el motor de la base de datos

que permite acceder a los datos, bloquearlos y modificarlos, y el esquema de la base de datos, que define la estructura lógica de la base de datos. Estos tres elementos fundamentales ayudan a proporcionar concurrencia, seguridad, integridad de datos y procedimientos uniformes de administración de datos [18].
En la presente investigación se estará utilizando como sistema gestor de base de datos Oracle.

Oracle 11g

Oracle es un sistema de gestión de base de datos relacional (RDBMS por el acrónimo en inglés, *Relational Database Management System*), desarrollado por Oracle *Corporation*, es considerado uno de los más potentes a nivel mundial. Ha incorporado en su sistema el modelo objeto relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. Así ofrece un servidor de bases de datos híbrido. Es uno de los más conocidos y ha alcanzado un buen nivel de madurez y de profesionalidad. Se destaca por su soporte de transacciones, estabilidad y escalabilidad. Los tipos de objetos de Oracle son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos [19].
Entre las principales ventajas de Oracle se destacan las siguientes [19]:

- Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia.
- Los programadores de aplicaciones pueden acceder directamente a tipos de objetos.
- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan lascaracterísticas del paradigma orientado a objetos.
- Tiene buen rendimiento y hace buen uso de los recursos.
- Posee un rico diccionario de datos.
- Brinda soporte a la mayoría de los lenguajes de programación.
- Es un sistema multiplataforma, disponible en Windows, Linux y Unix.

Servidor web Apache 2.4

Apache es un servidor web de código libre para la transferencia de hipertextos (HTTP por sus siglas en inglés de *Hypertext Transfer Protocol*,) para plataformas Unix, Windows. Su implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales [20].

Principales características [20]:

- Es un servidor web conforme al protocolo HTTP.
- Soporta tanto host basados en IP (por sus siglas en inglés *Internet Protocol*) como hosts virtuales.
- Apache soporta autentificación básica basada en la web.
- Modular, puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, para el desarrollo de módulos específicos.
- Extensible, gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.

Herramienta para crear prototipos de interfaz Axure 7

Es una herramientaorientada a diseñarprototipos básicos o avanzados de forma fácil dirigido a aplicaciones web y de escritorio. La ventana de la aplicación Axure RP Pro está dividido en 6 áreas principales [21]:

- Mapa del sitio-lista de jerarquía de las páginas.
- Reproductores.
- Maestros(Plantillas reutilizables).
- Área de las páginas (la principal área del diseño).
- Las notas de las páginas y de las iteraciones.
- Widget de anotaciones e iteraciones.

Para la realización de los prototipos de interfaz del presente trabajo de diploma se utilizará Axure RP Pro en su versión 7.0.

Herramienta de control de versiones Git

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta muy grandes con rapidez y eficiencia. Permite tener múltiples sucursales locales que pueden ser completamente independientes entre sí. La creación, fusión y eliminación de esas líneas de desarrollo lleva segundos. Esto significa que se pueden hacer cosas como [22]:

- Crear ramas y hacer fusiones entre las diferentes ramas.
- Reescribir historiales de repositorios.
- Añadir archivos de repositorios locales y confirma los cambios realizados.

- Enviar los cambios a la rama principal.
- Realiza cambios en los archivos con una herramienta de alojamiento de Git y lo confirma.
- Extrae los cambios.
- Crea ramas, donde se realiza algún cambio y lo confirma.
- Fusiona estas ramas con la rama principal.

Para el desarrollo de la presente investigación se decide utilizar esta herramienta de control de versiones debido a que es la utilizada en el proyecto GINA al cual pertenece el módulo a implementar. Se estará trabajando en su versión 11.10.4.

1.5. Diseño arquitectónico

El diseño arquitectónico se interesa por entender cómo debe organizarse un sistema y cómo tiene que diseñarse la estructura global de ese sistema. En el modelo del proceso de desarrollo de software, el diseño arquitectónico es la primera etapa en el proceso de diseño del software. Es el enlace crucial entre el diseño y la ingeniería de requisitos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos. La salida del proceso de diseño arquitectónico consiste en un modelo arquitectónico que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación [23].

Para la definición de la arquitectura a utilizar en el desarrollo de la solución se tuvo en cuenta la establecida en el desarrollo de GINA, donde se emplea la arquitectura que utiliza el marco de trabajo que se definió anteriormente, Symfony. Este marco de trabajo está basado en un patrón clásico del diseño web conocido como arquitectura Modelo-Vista-Controlador (MVC), que está formado por tres niveles [23]:

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

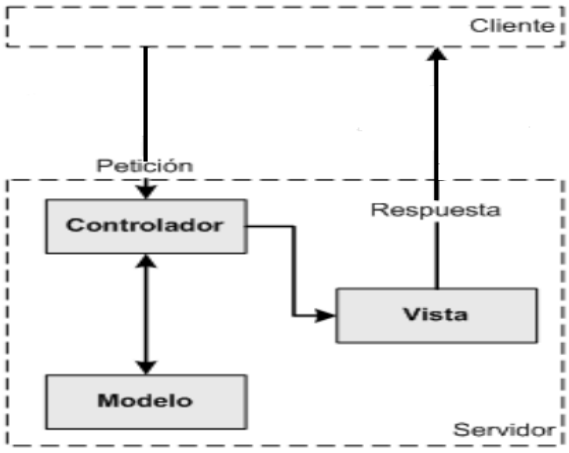

Figura 2. Arquitectura MVC

Fuente: Symfony 1.2 la guía definitiva

1.6. Técnicas de captura de requisitos

El objetivo de dichas técnicas es la obtención de requisitos. Se eligen según el proyecto a desarrollarse [24].

- Entrevistas y cuestionarios.
- Tormenta de ideas.
- Prototipos.
- Casos de uso.

En la presente investigación se utilizan como técnicas para obtener los requisitos, la entrevista y prototipos.

Entrevista:consiste en laformulación de preguntas relacionadas con varios aspectos de sistemas, sobre el proceso a informatizar y su funcionamiento. Se les realizan ausuarios de sistemas existentes, futuros usuarios, gerentes, empleados. Son útiles las preguntas abiertas para entender la problemática existente e identificar todos los aspectos necesarios para el desarrollo de las aplicaciones. Para la visualización de dicha entrevista ver anexos 1.

Prototipos:es un modelo del software que debe ser construido,primeramente,se capturan requisitos, se definen los objetivos globales, se diseña “rápidamente”centrándose en la parte visible al usuario (ej: entradas y salidas), se construye el prototipo y por último el cliente y el usuario evalúan el prototipo, refinando los requisitos.

1.7. Técnicas de validación de requisitos

La validación de requisitos es el proceso de verificar que los requisitos definan realmente el sistema que en verdad quiere el cliente. Se traslapa con el análisis, ya que se interesa por encontrar problemas con los requisitos. La validación de requisitos es importante porque los errores en un documento de requisitos pueden conducir a grandes costos por tener que rehacer, cuando dichos problemas se descubren durante el desarrollo del sistema o después de que éste se encuentra en servicio [24].

Hay algunas técnicas de validación de requisitos que se usan individualmente o en conjunto con otras:

- **Revisiones de requisitos:**los requisitos se analizan sistemáticamente usando un equipo de revisores que verifican errores e inconsistencias.
- **Creación de prototipos:**en esta aproximación a la validación, se muestra un modelo ejecutable del sistema en cuestión a los usuarios finales y clientes. Así, podrán experimentar con este modelo para constatar si cubre sus necesidades reales.
- **Generación de casos de prueba:**los requisitos deben ser comprobables. Si las pruebas para los requisitos se diseñan como parte del proceso de validación, esto revela con frecuencia problemas en los requisitos. Si una prueba es difícil imposible de diseñar, esto generalmente significa que los requisitos serán difíciles de implementar, por lo que deberían reconsiderarse. El desarrollo de pruebas a partir de los requisitos del usuario antes de escribir cualquier código es una pieza integral de la programación extrema.

Se estará utilizando como técnica las revisiones de requisitos, y además se generarán casos de pruebas para validar el correcto funcionamiento del módulo, el resultado de aplicarla se estará explicando en el capítulo 3 de la presente investigación.

Conclusiones del capítulo

El análisis de los principales conceptos asociados al dominio del problema permitió comprender los elementos teóricos que sustentan la presente investigación, así como el funcionamiento de los procesos que se desean informatizar en la AGR. El estudio acerca de los sistemas que anteriormente eran usado en la Aduana constató la necesidad de informatizar el proceso de selección de personas.La caracterización de la metodología de desarrollo de software, las herramientas, tecnologías y lenguajesempleados en GINA, permitió profundizar los conocimientos acerca de las mismas y comprender el porqué de su selección para una posterior implementación del módulo propuesto.Además, se expuso

una descripción de las técnicas de captura y validación de requisitos que serán aplicadas en próximoscapítulos.

Capítulo 2:Análisis, diseño e implementación del módulo propuesto

En el presente capítulo se definen los principales elementos a tener en cuenta para la implementación del módulo propuesto. En el mismo se realiza una descripción general, se realizael modelo conceptual y el diagrama de proceso de negocio. Se lleva a cabo el proceso de captura de los requisitos funcionales (RF) y no funcionales (RnF) aplicables para la implementación, la descripción de requisitos por procesos, se realiza además el diseño de la arquitectura, el modelo de datos y el empleo de patrones de diseño para una mejor implementación. Se establecen los estándares de codificación para llevar a cabo una implementación genérica y fácil de entender para un posterior mantenimiento.

2.1. Descripción de módulo propuesto

El Sistema de Gestión Integral de Aduanas (GINA), es una aplicación web, que gestiona los procesos que se llevan a cabo en la AGR, constituida por 9 subsistemas. Entre ellos el de Control de Personas, que gestiona la información de todos los pasajeros que entran y salen del país. Hoy en día este subsistema no cuenta con una funcionalidad que permita la selección de posibles pasajeros infractores sospechosos a cometer algún hecho delictivo. Este proceso se realiza de forma manual y empírica, por parte del analista de Aduana y debido a la gran cantidad de información que se maneja provoca un retraso en el proceso y omisión de posibles infractores.

En consecuencia, con lo anteriormente descrito,el módulo propuesto formará parte de GINA, específicamente del subsistema Control de Personas, el cual permitirá que el tiempo de análisis de los pasajeros en la AGR sea menor, el trabajo por parte del analista de Aduana sea más efectivo y se mantenga la protección en frontera. Su función será seleccionar pasajeros que cumplan con criterios de selección determinados, los cuales pasarán a formar parte de un perfil de riesgo ya creado previamente por el oficial de enfrentamiento y luego aplicado a un EAPI. Estos perfiles de riesgo son creados a partir de la experiencia acumulada ante los hechos delictivos,además pueden ser modificados y cancelados.Para su creación se tienen en cuenta datos como:

- Fecha de inicio del perfil de riesgo.
- Descripción del perfil de riesgo.
- Línea de enfrentamiento.
- Modalidad.
- Indicadores (Están conformados por un operador y un valor).

- Criterios (Son criterios de selección definidos por la AGR).
- Medidas a tomar.

La fecha de inicio del perfil de riesgo está dada en el momento que se comienza a utilizar el perfil de riesgo. La descripción del perfil depende de la función que realice el mismo, la línea de enfrentamiento son valores definidos por la AGR, y de ella depende la modalidad, valores también definidos por la AGR. Los criterios, son valores igualmente definidos por los cuales se realizará la selección de pasajeros, los indicadores también definidos estáncompuestos por un operador de riesgo y un valor que es en dependencia del criterio de selección. Las medidas a tomar también definidas por la AGR son en dependencia de la función del perfil.Una vez creados los perfiles de riesgo se procede a aplicar los mismos al listado de pasajeros de un vuelo, si la persona cumple con los criterios de selección de dicho perfil es seleccionada y luego marcada por el analista de Aduana del área de LCF, para realizarle el seguimiento y aplicar las medidas correspondientes al perfil.

2.2. Modelo conceptual

El modelo conceptual describe cómo se relacionan los conceptos del negocio necesarios para el entendimiento del mismo, representándolo de manera gráfica.

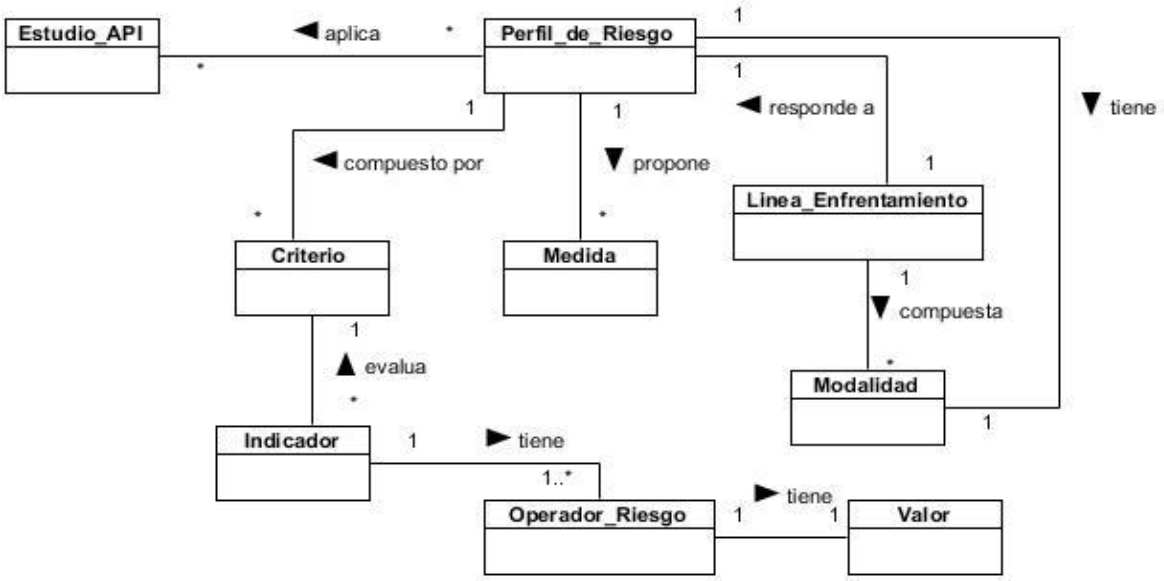

Figura 3. Modelo conceptual

Fuente: Elaboración propia

Un perfil de riesgo está compuesto por varios criterios de selección, estos criterios a su vez, son evaluados por varios indicadores que tienen varios operadores de riesgo y a estos se le asigna un valor. Un perfil de riesgo puede proponer una o varias medidas, además responde a una línea de enfrentamiento que está compuesta por una modalidad que debe tener el perfil de riesgo. Por último, los perfiles creados se aplican a los estudios API que se realizan.

2.3. Modelado de proceso de negocio

Con el diagrama de proceso de negocio (DPN)se realiza la descripción de las actividades que se llevan a cabo mediante el proceso de selección de pasajeros en la AGR. Se representan subprocesos, tales como la gestión de un indicador de riesgo y la gestión de perfiles, tareas y actividades que realizan las personas encargadas de dicha función en la AGR; el analista de Aduana del área de LCF y el oficial del área de enfrentamiento. De forma tal que se describa el trabajo realizado. Posibilita además representar y visualizar el funcionamiento del módulo de Selectividad en general, facilitando su entendimiento y haciendo posible la automatización del mismo y mejora.

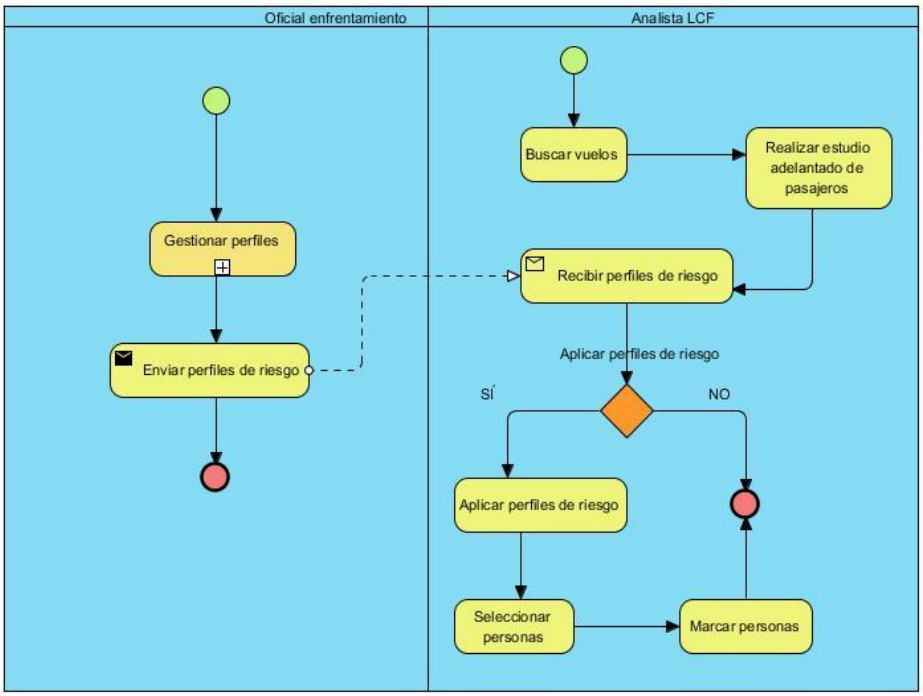

Figura 4. DPN Selectividad

Fuente: Elaboración propia

Capítulo 2: Análisis, diseño e implementación del módulo propuesto

2.4. Requisitos del módulo propuesto

Los requisitos de un software son descripciones de lo que el sistema debe hacer, el servicio que ofrece y las restricciones en su operación [24]. A continuación, se muestran los requisitos funcionales y no funcionales del módulo propuesto.

2.4.1. Requisitos funcionales

Los requisitos funcionales (RF) de un software son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos casos, los requisitos funcionales también explican lo que no debe hacer el sistema [24].

Para la captura e identificación de los requisitos funcionales se llevó a cabo la técnica entrevista (ver anexo 1) con el cliente obteniéndose un total de 16 requisitos funcionales para el desarrollo del módulo, los cuales se muestran en la siguiente tabla.

Tabla 2. Requisitos funcionales del módulo propuesto

No	Nombre	Descripción	Prioridad para el cliente	Complejidad
RF_1	Introducir perfil de riesgo.	Permite registrar un nuevo perfil de riesgo en el sistema.	Alta	Media
RF_2	Modificar perfil de riesgo.	Permite modificar la información de un perfil de riesgo.	Alta	Alta
RF_3	Cancelar perfil de riesgo.	Permite cancelar un perfil de riesgo.	Alta	Media
RF_4	Aplicar perfil de riesgo.	Permite aplicar un perfil de riesgo al listado de pasajeros.	Alta	Alta
RF_5	Mostrar usuarios que gestionen perfiles de riesgo.	Permite mostrar los usuarios que modifiquen, cancelen o creen perfiles de riesgo.	Alta	Alta

RF_6	Mostrar personas por perfiles de riesgo.	Permite mostrar las personas que han cumplido con determinado perfil de riesgo.	Alta	Alta
RF_7	Mostrar personas marcadas por perfiles de riesgo.	Permite mostrar las personas que ha sido marcadas de acuerdo con los perfiles de riesgo.	Alta	Alta
RF_8	Mostrar versiones de perfiles de riesgo.	Permite mostrar las modificaciones que se les han realizado a un perfil de riesgo.	Media	Alta
RF_9	Mostrar comportamiento de perfiles de riesgo.	Permite mostrar la cantidad de veces que se ha aplicado el perfil.	Media	Alta
RF_10	Listar perfiles de riesgo pasados.	Permite mostrar perfiles de riesgo que han sido utilizados.	Media	Media
RF_11	Listar perfiles de riesgo cancelados.	Permite mostrar perfiles de riesgo que hayan sido cancelados.	Media	Media
RF_12	Listar perfiles de riesgo futuros.	Permite mostrar perfiles de riesgo que se aplicarán posterior a la fecha actual.	Media	Media
RF_13	Listar perfiles de riesgo vigentes.	Permite mostrar perfiles de riesgo que están siendo aplicados en la actualidad.	Media	Media
RF_14	Insertar indicador de riesgo	Permite adicionar un indicador al perfil de riesgo.	Alta	Alta
RF_15	Modificar indicador de riesgo	Permite modificar el operador o el valor de un indicador de riesgo existente en un perfil de riesgo.	Alta	Alta
RF_16	Eliminar indicador de riesgo	Permite eliminar un indicadorya existente en un perfil de riesgo.	Media	Media

Fuente: Elaboración propia

2.4.2. Requisitos no funcionales

Los requisitos no funcionales (RNF) de un software son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requisitos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema [24].

El sistema consta de 12 requisitos no funcionales descritos en la siguiente tabla.

Tabla 3. Requisitos no funcionales del módulo propuesto

Usabilidad	
RNF 1	DesarrollarunasoluciónwebintegradaalSistemade Gestión Integral de Aduanas.
RNF 2	El sistema debe presentar una interfaz sencilla que permita la fácil interacción y entendimiento de las actividades que realiza el usuario.
Seguridad	
RNF 3	El sistema debe solicitar una verificación sobre acciones irreversibles (cancelaciones de perfiles de riesgo).
Software	
RNF 4	Se requiere de un navegador web instalado en las computadoras clientes.
RNF 5	El lenguaje de programación a utilizar es PHP 5.4.
RNF 6	Como marco de trabajo se empleará Symfony 1.2.8.
RNF 7	Como Entorno de Desarrollo Integrado se empleará PhpStorm 2017.2.
RNF 8	Como servidor web se utilizará Apache 2.4.
RNF 9	El Sistema Gestor de Bases de Datos utilizará Oracle 11g.

RNF 10	El diseño de la base de datos se realizará con Visual Paradigm 8.0.
Hardware	
RNF 11	El sistema requiere de una computadora que haga la función de servidor, la cual debe tener una memoria RAM de 1 GB o más y capacidad de disco duro de 50 GB o más.
Confiabilidad	
RNF 12	El sistema no debe permitir entrada de datos incorrectos y debe notificar los errores.

Fuente: Elaboración propia

2.5. Descripción de requisitos por proceso

La descripción de requisitos por procesos es el resultado de aplicar o establecer el escenario número 3 de la metodología AUP-UCI, para representar un mayor nivel de detalle y las relaciones entre los procesos identificados en el negocio. Establecido previamente por los analistas del proyecto debido a que el negocio a informatizar presentaba una total independencia de los procesos que se manejaban a las personas encargadas que lo ejecutaban, proporcionando además objetividad, solidez, y su continuidad. Para la visualización de todas las descripciones de requisitos debe consultar el expediente de proyecto Selectividad.

Descripción textual del requisito

Tabla 4. Descripción del RF Introducirperfil de riesgo

Precondiciones	El usuario está autenticado y tiene los permisos correspondientes para realizar esta acción.
Flujo de eventos	
Flujo básico introducir perfil de riesgo	
1.	Selecciona la opción Estudio API, gestionar perfiles EA del menú principal.
2.	Se selecciona la opción Adicionar.
3.	El sistema muestra una interfaz para introducir los datos del perfil de riesgo: • Fecha inicio • Descripción • Línea enfrentamiento

	• Modalidad • Indicadores • Medidas a tomar
4.	El usuario introduce los campos requeridos y acciona el botón Aceptar.
5.	El sistema genera un número de perfil.
6.	Muestra un mensaje confirmando el registro del perfil e indica el número asignado.
7.	Concluye el requisito.
Pos-condiciones	
1.	Queda registrado el perfil de riesgo.
Flujos alternativos	
1	N/A
Pos-condiciones	
1	N/A
Validaciones	
1	El número del perfil está compuesto por un número consecutivo de tres cifras. Las cifras a la izquierda se rellenan con cero.
2	Todos los campos son obligatorios.
3	Para cada indicador se introduce operador y valor. En caso de las fechas se introduce un rango de valores.
Conceptos	N/A
Requisitos especiales	N/A
Asuntos pendientes	N/A

Fuente: Elaboración propia

Prototipo de interfaz gráfica de usuario

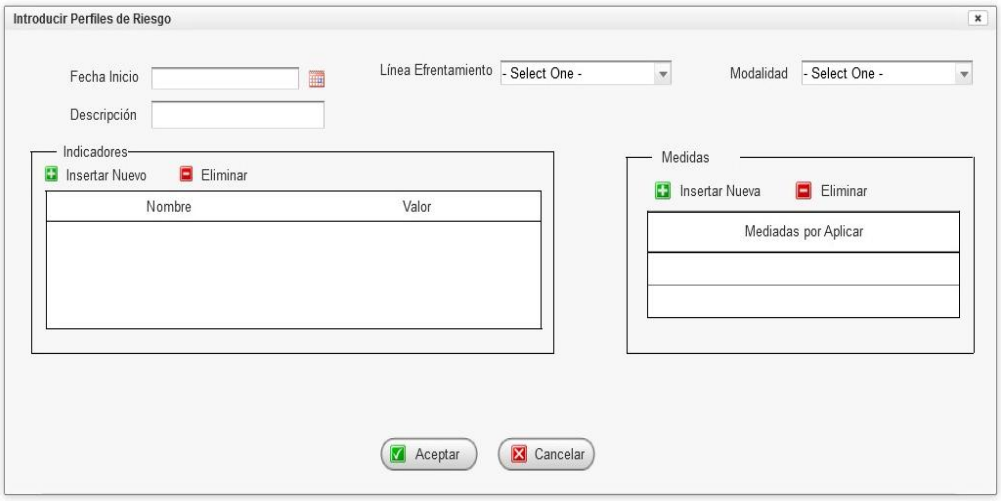

Figura 5. Interfaz de usuario introducirperfil de riesgo

Fuente: Elaboración propia

Figura 6. Interfaz de usuario adicionar indicador

Fuente: Elaboración propia

2.6. Validación de los requisitos

Para la realización de la validación de los requisitos se utilizaron los criterios que propone el proceso de desarrollo de software con enfoque ágil basado en el nivel 2 de CMMI, donde se responden a varias interrogantes que decide si el requisito se aprueba o no.

Interrogantes para validar los requisitos del cliente

- ¿El proveedor del requisito es un proveedor válido?
- ¿El requisito es modificable?
- ¿El requisito no es ambiguo?
- ¿El requisito es congruente con otros requisitos relacionados?
- ¿El requisito puede ser implementado?
- ¿El requisito puede ser probado?
- ¿El requisito es correcto?
- ¿El requisito es traceable?

Con la aplicación de estos criterios para validar la aprobación de los requisitos se obtuvo como resultado el 100% de los requisitos aprobados como se muestra en el documento “Criterios para validar requisitos del cliente” que se encuentra en el expediente del módulo Selectividad.

Tras aplicar la técnica de revisiones de requisitos descrita en el capítulo 1, se realizaron verificaciones en el documento “Especificaciones de requisitos de Selectividad”. Este proceso comprendió:

- Verificaciones de validez: los requisitos deben cumplir con las necesidades del cliente. Luego de realizar el análisis de los requisitos capturados surgieron funciones adicionales y cambios en los que ya estaban identificados.
- Verificaciones de consistencia: los requisitos no deben contradecirse en las especificaciones escritas, no debe haber restricciones o descripciones que estén opuestas a las reglas definidas.
- Verificaciones de completitud: los requisitos deben incluir todas las funcionalidades propuestas por el cliente, satisfacer de manera general todas las necesidades acordadas.
- Verificabilidad: para evitar posibles discusiones entre los miembros del equipo del proyecto y el cliente, se revisó que los requisitos estuvieran descritos de manera que puedan ser verificables, o sea, que se puedan diseñar casos de pruebas orientados a estos y que demuestren que el sistema a entregar responde a las necesidades del cliente.

Como resultado de este proceso se identificaron inconsistencias en las especificaciones tales como:

- Errores ortográficos.
- Componentes necesarios que no se habían incluidos.
- Requisitos pocos detallados.

- Falta de concordancia entre la complejidad de la especificación de requisitos y la registrada en el documento de Evaluación de requisitos.

2.7. Modelo de datos

El modelo de datos hace referencia a cada una de las tablas necesarias para la realización del módulo propuesto, así como las relaciones entre ellas. Su representación fue realizada a través del diagrama Entidad-Relación, mostrando las entidades, atributos y relaciones entre estas. El mismo está compuesto por 15 tablas, donde las principales son: la tabla *LCF_PERFIL_RIESGO* es donde se gestionan todos los elementos del negocio. También es la encargada de que exista una relación entre las demás tablas, la tabla *PERFIL_RIESGO_PERSONA* es la encargada de establecer la relación entre el perfil creado y la persona a la cual se le va a aplicar, la tabla *TC_MEDIDAS_A_TOMAR* es la contenedora de las acciones a realizar sobre aquellas personas marcadas por un perfil, la tabla *TC_INDICADOR_RIESGO* contiene aquellas características asociadas a los pasajeros, por las cuales los analistas se van a regir para su criterio de selección, la tabla *TC_OPERADOR_RIESGO* contiene los operadores de comparación que ayudan a conformar un criterio de selección, en la tabla *TC_LINEA_DE_ENFRENTAMIENTO* es por donde se va a aplicar el perfil, la cual depende de la modalidad seleccionada en la tabla *TC_MODALIDAD.*

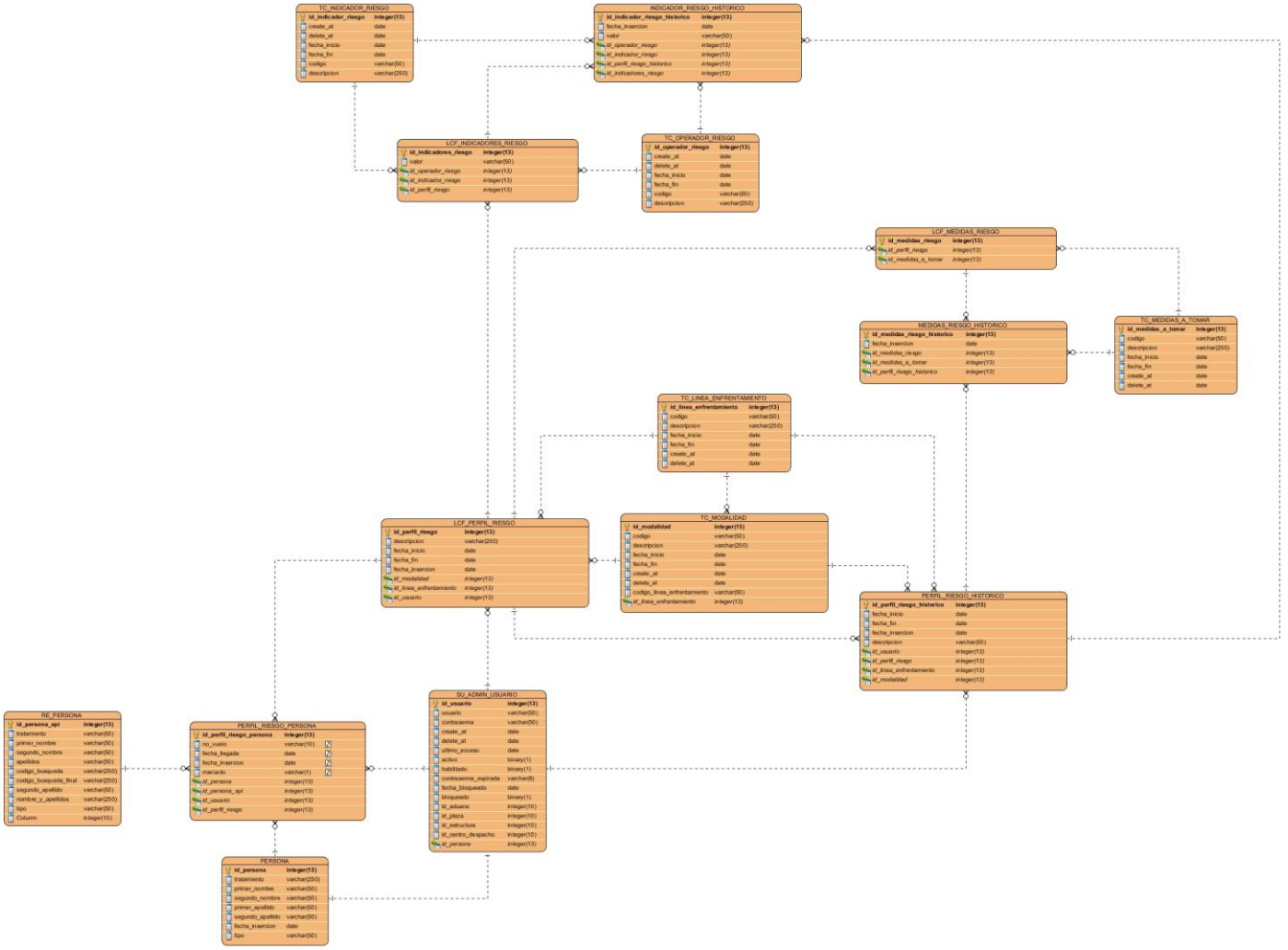

Figura 7. Modelo de datos

Fuente: Elaboración propia

2.8. Patrones del diseño

Los patrones GRASP (General Responsibility Asignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones [20]. Para el diseño del módulo se emplearon los siguientes patrones GRASP.

Bajo acoplamiento

El bajo acoplamiento dentro de un sistema indica que los módulos no conocen o conocen muy poco el funcionamiento interno de otros módulos, evitando la fuerte dependencia entre ellos [27].En la presente investigación, se ve reflejado en el manejo de las funcionalidades de las clases, donde *MarcarPersonaPorPerfil* es la funcionalidad de la controladora que emite el usuario, *PerfilRiesgo* es quien gestiona la información de los perfiles, *AsociarPerfilPersona* es la función encargada de vincular a los pasajeros con los perfiles.

Figura 8. Aplicación del patrón bajo acoplamiento

Fuente: Elaboración propia

Alta cohesión

La alta cohesión se refiere la medida de qué tanto un módulo de un sistema tiene una sola responsabilidad. Por lo tanto, con alta cohesión será aquel que guarde una relación entre sus funcionalidades, manteniendo un enfoque a su único propósito [27]. En la presente investigación, se muestra el bajo nivel de sobrecarga de responsabilidades.

Figura 9. Aplicación de patrón alta cohesión

Fuente: Elaboración propia

Controlador

El patrón controlador es encargado de representar al sistema completo o a la organización. Delega la responsabilidad en otras clases con las que mantiene un modelo de alta cohesión [24]. En la presente investigación, se evidencia en la clase controladora PerfilRiesgo, encargada de controlar el negocio.

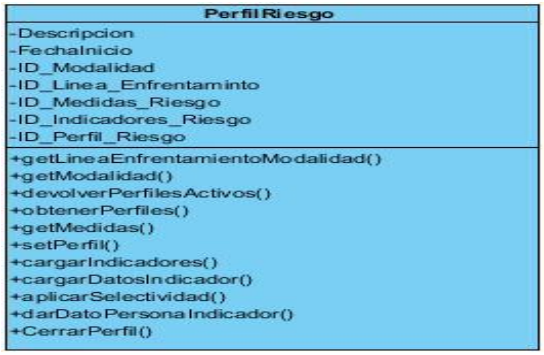

Figura 10. Aplicación del patrón controlador

Fuente: Elaboración propia

Creador

El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Su propósito fundamental es encontrar un creador, debe conectarse con el objeto producido en cualquier evento [24]. En la presente investigación se evidencia en la clase *Action.class.php*. El patrón se aplica en el momento de crear nuevas instancias de *PerfilRiesgoHistorico*, tal y como ilustra en el área señalada.

```
class PerfilRiesgoHistoricoPeer extends BasePerfilRiesgoHistoricoPeer {

    public static function InsertarPerfilHistorico(LcfPerfilRiesgo $perfil, $con)
    {
        if($perfil instanceof LcfPerfilRiesgo){
            $fNow = new DateTime();//Fecha insercion empleada por historico!
            $perfilHist = new PerfilRiesgoHistorico();
            $perfilHist -> setFInsercion($fNow->format( format: 'd-m-Y H:m:s'));
            $perfilHist -> setIdPerfilRiesgo($perfil->getIdPerfilRiesgo());
            if(is_null($perfil->getFFin()))
                $perfilHist -> setFInicio($perfil->getFInicio( format: 'd-m-Y'));
            else
                $perfilHist -> setFInicio($perfil->getFFin( format: 'd-m-Y H:m:s'));
//              $perfilHist -> setFFin(new DateTime('9999-12-31'));
            $perfilHist -> setIdLineaEnfrentamiento($perfil->getIdLineaEnfrentamiento());
            $perfilHist -> setIdModalidad($perfil->getIdModalidad());
            $perfilHist -> setDescripcion($perfil->getDescripcion());
            $perfilHist -> save($con);
            return $perfilHist;
        }else
            throw new Exception( message: 'Perfil incorrecto.');
    }
```

Figura 11. Aplicación de patrón creador

Fuente: Elaboración propia

Como otros patrones empleados están los patrones GOF (por sus siglas en inglés *Gang of Four*)o conocido también como patrones de la pandilla de los cuatros.Los mismos son unas técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Para el diseño del módulo se emplearon los siguientes patrones GOF.

Singleton

Recibe su nombre debido a que sólo se puede tener una única instancia para toda la aplicación de una determinada clase [24]. En la presente investigación, se evidencia en la asignación de un *id* a cada clase, el cual va a ser único e inmodificable.

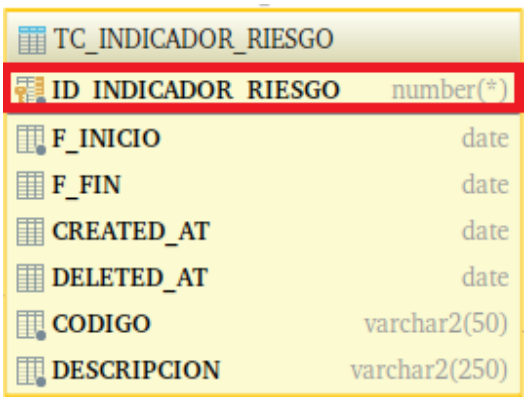

Figura 12. Aplicación del patrón singleton

Fuente: Elaboración propia

Fachada

Se requiere una interfaz común para un conjunto de implementaciones o interfaces dispares [24]. En la presente investigación, se evidencia con el uso del panel de diseño *gridPanel*para la elaboración de tablas en el desarrollo de la solución.

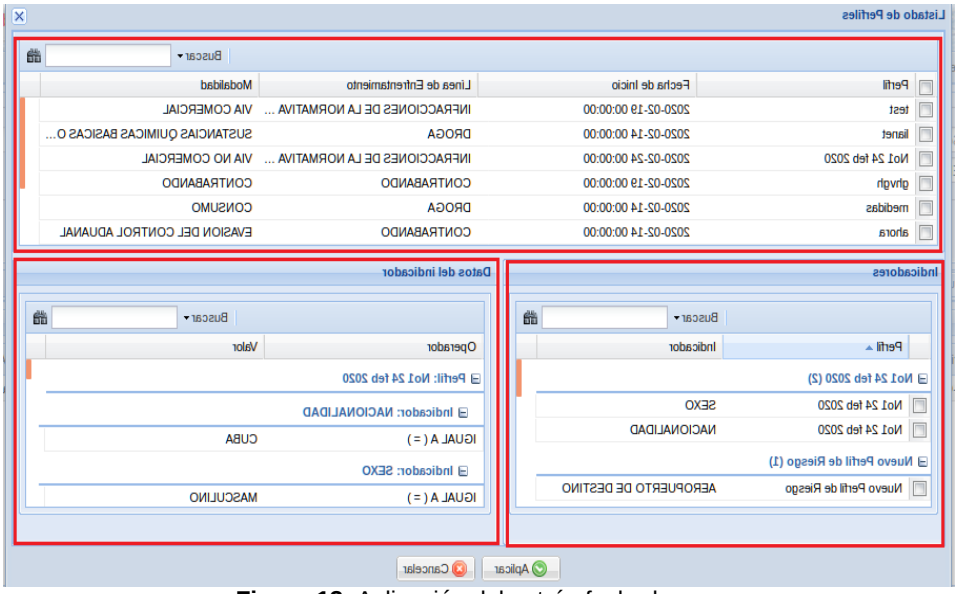

Figura 13. Aplicación del patrón fachada

Fuente: Elaboración propia

2.9. Estándares de codificación

Los estándares de código, son parte de las llamadas buenas prácticas o mejores prácticas, estas son un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y las cuales, bien aplicadas pueden incrementar la calidad del código, notablemente. El objetivo de crear un estándar de codificación es que los programadores se rijan por el mismo a la hora de implementar. Este estándar debe servir para que el personal de un proyecto identifique de forma sencilla cuál es el objetivo y las funcionalidades que brinda cada una de las clases, funciones y demás componentes de software dada su nomenclatura, es necesario que esto se pueda identificar a simple vista. Además, debe servir de guía para los implementadores que tienen que continuar el desarrollo de las aplicaciones posteriormente [28]. Para la presente investigación se utilizan estándares de códigos propios del lenguaje de programación y la arquitectura regida por la utilización del marco de trabajo arquitectónico Symfony.

Reglas generales

Cuando se incluyen abreviaturas en mayúsculas no se debe incluir el nombre completo, sino que se utiliza el primer nombre en mayúscula y el resto en minúsculas.

Aplicaciones

- Las aplicaciones deben tener nombres que dejen reflejado claramente cuál es el propósito de la misma, ya en una palabra o siglas.
- Se debe evitar mientras sea posible la utilización de palabras compuestas o de varias palabras, en caso de que sean palabras compuestas se utilizará la notación **UpperCamelCase[14].**

Módulos

- Deben referirse a los nombres de tablas en caso de que se trate de un módulo generado por el CRUD[15].
- En caso de que sean módulos del negocio de la aplicación debe cumplir con las mismas reglas

[14]UpperCamelCase: Es una convención de nomenclatura en la que un nombre está formado por varias palabras que se unen como una sola palabra con la primera letra de cada una de las múltiples palabras en mayúscula dentro de la nueva palabra que forma el nombre.

[15]CRUD: Resume las funciones requeridas por el usuario para crear y gestionar datos. CRUD hace referencia a un acrónimo en el que se reúnen las primeras cuatro operaciones fundamentales de aplicaciones persistentes, crear(créate), leer (read), actualizar (update), eliminar (delete).

de codificación de los nombres de las aplicaciones.

Acciones

Symfony trae una propia nomenclatura para las clases y funciones, pero no especifica claramente cuál es el nombre que se le debe de poner a cada una de las funcionalidades del modelo que serán accedidas por el usuario.

- Dentro de las especificaciones del marco de trabajo está que cada una de estas acciones debe comenzar con la palabra **execute**[16].
- Todos los nombres de acciones deben estar en la nomenclatura "**CamelCase**"[17]comenzando por la palabra **execute**.
- En caso de ser acciones referentes a un módulo de CRUD de una tabla deben ser nombresespecíficos como **executeNuevo**, **executeEditar**.

```
public function executeAplicarSelectividadEAPI(sfWebRequest $request)
{
```

Figura 14. Uso de la nomenclatura CamelCase y uso de execute.

Fuente: Elaboración propia

Nombre de las clases

- Los nombres de las clases deben estar expresados en notación UpperCamelCase.
- No se deben utilizar guiones bajos en el nombre "_".
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- En los nombres compuestos por más de tres palabras se debe revisar el diseño, no sea que se les estén dando a la clase más responsabilidades de las que realmente tiene.

```
public function executeAdicionarPerfilEA(sfWebRequest $request){
    $critSel =str_replace("'",'"',$request->getPostParameter('critSel',null));
    $critSel = json_decode($critSel);
    $medidas = json_decode($request->getPostParameter('medidas',null));
```

Figura 15. Creación de funciones con nombres sugerentes a su uso

Fuente: Elaboración propia

[16]Execute: Es un prefijo en la nomenclatura que define Symfony para los métodos del controlador.
[17]CamelCase: Es una convención de nomenclaturas compuesto por un número de palabras unidas sin espacios con letra inicial de cada palabra en mayúscula.

- Los métodos deben estar adecuadamente comentariados para facilitar un posterior mantenimientoo reutilización.

```
/**
 * En esta funcion se registran los perfiles de Riesgo para EA
 * @param sfWebRequest $request
 */
public function executeAdicionarPerfilEA(sfWebRequest $request){
    $critSel =str_replace("'",'"',$request->getPostParameter('critSel',null));
    $critSel = json_decode($critSel);
    $medidas = json_decode($request->getPostParameter('medidas',null));

    $fechaI = $request->getPostParameter('fechaI',null);
    $arregloFechaI=explode('/',$fechaI);
    $idModalidad = $request->getPostParameter('modalidad',null);
    $idLineaEnf = $request->getPostParameter('lineaE',null);
    $descripcion = $request->getPostParameter('descripcion',null);
```

Figura 16. Comentarios para facilitar el mantenimiento

Fuente: Elaboración propia

2.10. Descripción del módulo implementado

El módulo Selectividad informatiza el proceso de la selección de personas que cumplen con un perfil de riesgo determinado. Estos perfiles de riesgo son creados por el oficial de enfrentamiento encargado de esta función en la AGR, basados en criterios predefinidos de interés aduanal.

El sistema permite crear, modificar y cancelar estos perfiles, para luego ser aplicados a los pasajeros.

Figura 17. Interfaz de usuario perfiles de riesgo

Fuente: Elaboración propia

Aplicar Selectividad

Primeramente, se carga el API del vuelo enviado al analista de la Aduana, luego se procede a efectuar la selectividad de pasajeros. Cuando se aplica un perfil de riesgo sobre un API se obtiene como resultado un listado de pasajeros que aplican al perfil de riesgo, o sea las personas que cumplen con los criterios de selección definidos en el perfil de riesgo, siendo dicho listado registrado y asociado al analista encargado.

Figura 18. Interfaz de usuario realizar estudio API

Fuente: Elaboración propia

Figura 19. Interfaz de usuario asociada a aplicar perfil(es) estudio adelantado (EA)

Fuente: Elaboración propia

Por último, se elaboran un conjunto de reportes con el fin de obtener información estadística del comportamiento que se evidencia entre los infractores detectados, el comportamiento de los analistas que gestionan los perfiles, el comportamiento de los propios perfiles, determinando la efectividad de los mismos, permitiendo obtener información a partir de datos registrados.

Reportes que genera el módulo Selectividad

1. Personas marcadas por perfiles de riesgo

Permite obtener un listado de las personas marcadas por perfiles de riesgo.

Funcionalidad:

- Permite obtener datos de las personas que han aplicado a los perfiles de riesgo.
- Permite obtener datos de las personas que han aplicado a los perfiles de riesgo y luego fueron marcadas.
- Permite obtener datos de las personas que han aplicado a los perfiles de riesgo y no fueron marcadas.

Filtros:

Fecha inicio.

Fecha fin.

Perfil (Listado de los perfiles vigentes).

Estado (Solo vigentes o históricos).

Línea de enfrentamiento.

Modalidad.

Resultados a mostrar (Muestra los marcados, incluye sin marcar o todas las personas).

3. Versiones de los perfiles de riesgo

Permite obtener un listado con las modificaciones que se les ha realizado a los perfiles de riesgo.

Funcionalidad:

- Permite mostrar todos los perfiles creados hasta la actualidad.
- Permite mostrar dado el nombre de un perfil, todas las versiones creadas del mismo.
- Permite mostrar dado un rango de fecha, el comportamiento de los perfiles.

Filtros:

Nombre del perfil.

Fecha inicio.

Fecha final.

4. Estadísticas del comportamiento por perfiles de riesgo

Permite obtener un listado con datos estadísticos sobre los perfiles y su aplicación en el API.

Funcionalidad:

- Permite obtener un listado con datos estadísticos sobre los perfiles de riesgo aplicados en un período dado.
- Permite obtener un desglose de los usos hechos del perfil con la información estadística referente a los mismos.

Filtros:

Fecha inicio.

Fecha fin.

Perfil (Listado de los perfiles vigentes).

Línea de enfrentamiento.

Modalidad.

Conclusiones del capítulo

La descripción general del módulo propuesto facilitó la comprensión del proceso de negocio a informatizar. A través de la descripción de requisitos por proceso se pudo obtener una mejor documentación de cada uno de los procesos que componen el módulo propuesto, describiendo todos los requisitos obtenidos a partir de la aplicación de la técnica de obtención de requisitos. El diseño del modelo de datos estableció las relaciones entre las tablas más importantes que se emplean en la implementación.El uso de patrones del diseño llevó a cabo una correcta implementación permitiendo la reutilización del módulo en aplicaciones futuras.Además, la utilización de los estándares de codificación, permitió una implementación generalizada para un posterior mantenimiento del módulo.

Capítulo3:Validación del módulo propuesto

En el presente capítulo se llevaa cabo la validación del módulo implementado, aplicando las técnicas de validación de requisitos definida en el capítulo 1 y llevando a cabo las pruebas de software definidas. Se tendrá en cuenta una estrategia de prueba que guiará el proceso y podrá comprobar el correcto funcionamiento del módulo desarrollado, verificando que estén implementados todos los requisitos identificados y que se realicen correctamente.

3.1. Pruebas de software

Las pruebas de softwarerepresentan el porcentaje más grande de esfuerzo técnico en el proceso de software. Sin importar el tipo de software que se construya, una estrategia para planificar, ejecutar y controlar pruebas sistemáticas comienza por considerar pequeños elementos del software y moverse hacia afuera, hacia el programa como un todo.El objetivo de las pruebas del software es descubrir errores.[25].

Niveles de pruebas

Un nivel de prueba es un grupo de actividades que se organizan y administran juntas dentro dela ejecución de un proceso de pruebas que tiene como objetivo verificar y validar loscomponentes de un producto. Estos niveles permiten seleccionar diferentes tipos y técnicas depruebas a realizar en cada nivel [25].

En la presente investigación se llevarán a cabo los siguientes niveles de pruebas:

Pruebas de unidad: se ponen a prueba unidades de programa o clases de objetos individuales. Las pruebas de unidad deben enfocarse en comprobar la funcionalidad de objetos o métodos. Tienen como objetivo verificar que los flujos de control, los flujos funcionales y los flujos de datos delelemento software son cubiertos, y que ellos funcionan como se espera [25].

Pruebas de integración: la prueba de integración es ejecutada para asegurar que los componentes software y hardware del producto operan adecuadamente cuando interactúan entre ellos para ejecutar un caso de uso (o transacción de negocio). Las pruebas de integración exponen la no compleción o los errores en las especificaciones de la interfaz de cada paquete software siendo integrado con los demás [25].

Prueba de aceptación: la prueba de aceptación de usuario es la prueba final antes de desplegar el software en los ambientes de operación. El objetivo de la prueba de aceptación es verificar que el software está listo, que puede ser utilizado, que satisface los criterios de aceptación, y que cubre aquellas necesidades y expectativas de los clientes para los cuales el software se construyó [25].

3.2. Estrategia de prueba para la validación del módulo propuesto

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategiade prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados [25].

En la presente investigación se decide realizar una estrategia de prueba que permita verificar que el módulo desarrollado cumple con las necesidades del cliente y que al mismo tiempo lo realiza eficientemente.La siguiente tabla muestra los niveles de prueba descritos anteriormente con las diferentes pruebas y las técnicasdefinidas.

Tabla 5. Estrategia de prueba para la validación del módulo

Pruebas	Tipo de prueba	Método	Técnica
Unitarias	Funcional	Caja blanca	Camino básico
		Caja negra	Particiones de equivalencia
Integración	Funcional	Caja negra	Incremental
Aceptación	Funcional	Caja negra	Alfa

Fuente: Elaboración propia

3.3. Pruebas de caja negra

Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requisitos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requisitos funcionales para un programa. Las pruebas de caja negra no son una alternativa para los métodosde caja blanca. En vez de ello, es un enfoque complementario que es probable que descubra una clase de errores diferente que los métodos de caja blanca [25].

Las pruebas de caja negra intentan encontrar errores en las categorías siguientes [25]:

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en las estructuras de datos o en el acceso a bases de datos externas.
- Errores de comportamiento o rendimiento.
- Errores de inicialización y terminación.

Para realizar las pruebas de caja negra existen varias técnicas, las cuales se explican a continuación:

Técnica de la partición de equivalencia:esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Técnica del análisis de valores límites:esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Técnica de grafos de causa-efecto:es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Para la validación del módulo se estará aplicando específicamente la técnica de partición de equivalencia,la cual se describe de la siguiente manera:

Partición de equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores [25].

El objetivo de estos casos de prueba es que sean efectivos descubriendo defectos en el módulo y se muestre que se satisfacen los requisitos identificados. Se puede apreciar el caso de prueba de caja negra aplicando la técnica antes descrita al requisito de introducir perfiles de riesgo en el sistema en el anexo 4.

Para consultar los diseños de casos de pruebas en su totalidad, se debe consultar el expediente del módulo Selectividad.

Al aplicar la prueba de caja negra con su respectiva técnica se obtuvieron los siguiente resultados en cada una de las iteraciones realizadas.

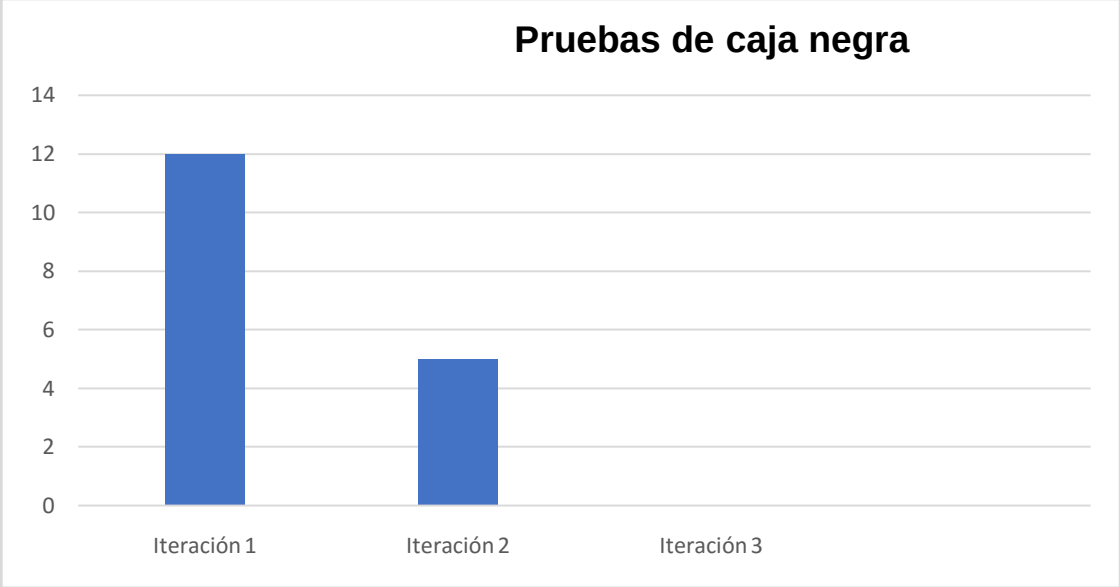

Figura 20. Cantidad de no conformidades detectadas en las pruebas de caja negra

Fuente: Elaboración propia

En la primera iteración se detectaron un total de 12 no conformidades (NC), 3 NC de errores de validación, 3 NC de funcionalidad, 2 NC de correspondencia entre la aplicación y el documento, 4 NC de errores ortográficos. En la segunda iteración se detectaron 5 no conformidades, dondeaún existían errores de funcionalidad y validación. Posteriormente en una tercera y última iteración no se detectaron no conformidades obteniéndose resultados satisfactorios.

3.4. Pruebas de caja blanca

La prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que [24]:

- Se garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez.
- Se revisen todas las decisiones lógicas en los lados verdadero y falso.
- Se ejecuten todos los bucles en las fronteras y dentro de las fronteras operativas.

- Se revisen estructuras de datos internas para garantizar su validez.

Para la aplicación de esta prueba en la presente investigación se utilizará la técnica de la **prueba del camino básico**, la cual consiste en garantizar que se pueda probar al menos una vez cada una de las sentencias del programa, partiendo de la obtención de la medida de la complejidad de un procedimiento o algoritmo y la obtención de un conjunto básico de caminos de ejecución de este, los cuales son utilizados para obtener los casos de prueba [25].

En la aplicación de la prueba se tuvo en cuenta la función darVigenteDadoIdPerfilRiesgo, a continuación, se muestran los pasos que se realizaron para realizar la técnica del camino básico:

```
public static function darVigenteDadoIdPerfilRiesgo($idPerfilRiesgo, $absolute = true,$perfil = null)
{
1   $crit = new Criteria();
    $crit -> add( p1: self::ID_PERFIL_RIESGO, $idPerfilRiesgo);
    $critF2 = $crit->getNewCriterion( column: self::F_FIN, value: null);
2   if(!$absolute && $perfil instanceof LcfPerfilRiesgo){
3       $critF1 = $crit->getNewCriterion( column: self::F_INICIO,$perfil->getFInicio(), comparison: Criteria::LESS_EQUAL);
4       $critF3 = $crit->getNewCriterion( column: self::F_FIN,$perfil->getFInicio( format: 'Y-m-d'), comparison: Criteria::GREATER_EQUAL);
        $critF2 -> addOr($critF3);
    }else{
5       $today = new DateTime();
        $critF1 = $crit->getNewCriterion( column: self::F_INICIO,$today->format( format: 'Y-m-d'), comparison: Criteria::LESS_EQUAL);
    }
6   $critF1 -> addAnd($critF2);
    $crit -> add($critF1);
    $result = self::doSelectOne($crit);
7   if($result instanceof PerfilRiesgoHistorico && !is_null($result->getFFin()))
8       $result = self::doSelect($crit);
9   return $result;
10}
```

Figura 21. FunciónderVigenteDadoIdPerfilRiesgo

Fuente: Elaboraciónpropia

1. Confeccionar el grafo de flujo: usando el código de la figura 23 se realizó la representación del grafo de flujo, el cual está compuesto por los siguientes elementos:

Nodos: son círculos que representan una o más sentencias procedimentales.

- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

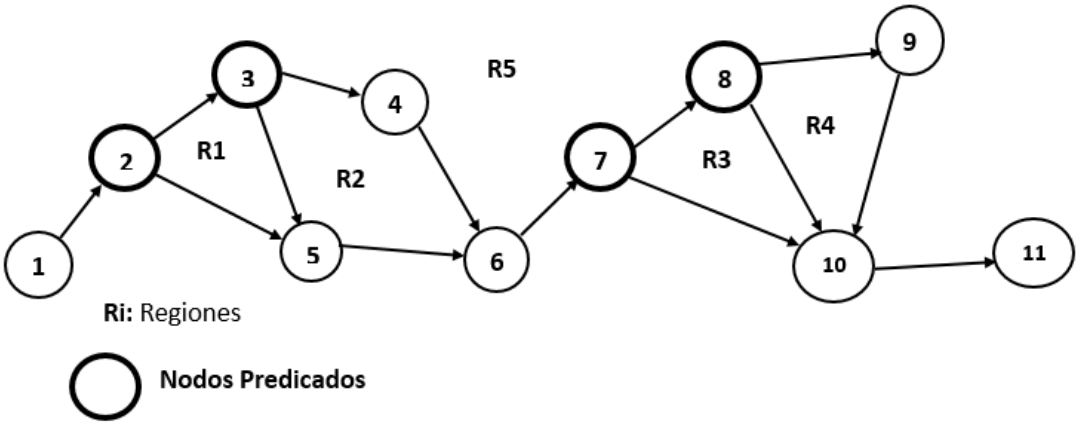

Figura 22. Grafo de flujo de la función darVigenteDadoIdPerfilRiesgo
Fuente:Elaboraciónpropia

2. Calcular la complejidad ciclomática: proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, la complejidad ciclomática se calcula de tres formas distintas. En la solución propuesta se aplican las tres variantes con el propósito de triangular los resultados obtenidos, validando que la cantidad de caminos a definir es la correcta:

Variante 1:

V(G)=E-N+2, donde E es el número de aristas del gráfico de flujo y N el número de nodos del gráfico de flujo G.

V(G)=14 – 11 + 2

V(G)= 5

Variante 2:

V(G)=P+1, donde P es el número de nodos predicados (nodos de los que parten 2 o más aristas) incluidos en el gráfico de flujo G.

V(G)= 4 + 1

V(G)=5

Variante 3:

Cantidad de regiones correspondientes al gráfico.

Teniendo en cuenta el número de regiones que presenta el grafo de flujo obtenido, se definen un total de 5 regiones.

Las tres variantes aplicadas brindan como resultado que el valor de la complejidad ciclomática del método darVigenteDadoIdPerfilRiesgo () es igual a 5.

3. Determinar un conjunto básico de caminos linealmente independientes: una vez aplicadas las tres variantes para calcular la complejidad ciclomática del método darVigenteDadoIdPerfilRiesgo(), se obtiene que el número de caminos linealmente independientes de la estructura de control del método es 3, definiéndose los siguientes caminos:

Camino básico #1: 1-2-5-6-7-10-11

Camino básico #2: 1-2-3-5-6-7-10-11

Camino básico #3: 1-2-3-4-6-7-10-11

Camino básico #4: 1-2-3-5-6-7-8-10-11

Camino básico #5:1-2-3-4-6-7-8-9-10-11

Generación de los casos de prueba:

Cada camino independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. En este caso se obtuvieron 5 caminos básicos, por tanto, se hace necesario la confección de igual número de casos de pruebas, para aplicar las pruebas a este método. A continuación, se muestra el caso de prueba diseñado para el camino básico # 2:

Tabla 6. Generación de caso de pruebas para el camino básico # 2

Caso de pruebas: Camino básico # 2	
Entrada	Se introduce la variable *$idPerfilRiesgo* por la cual se va a realizar la búsqueda
Resultados esperados	Se retorna la variable *$result*como un objeto
Condiciones	El método que se ejecuta en la consulta es un *doSelectOne*

Fuente: Elaboración propia

Al aplicar la prueba de caja blanca se alcanzaron los siguientes resultados en cada una de las iteraciones como se muestra en la siguiente figura.

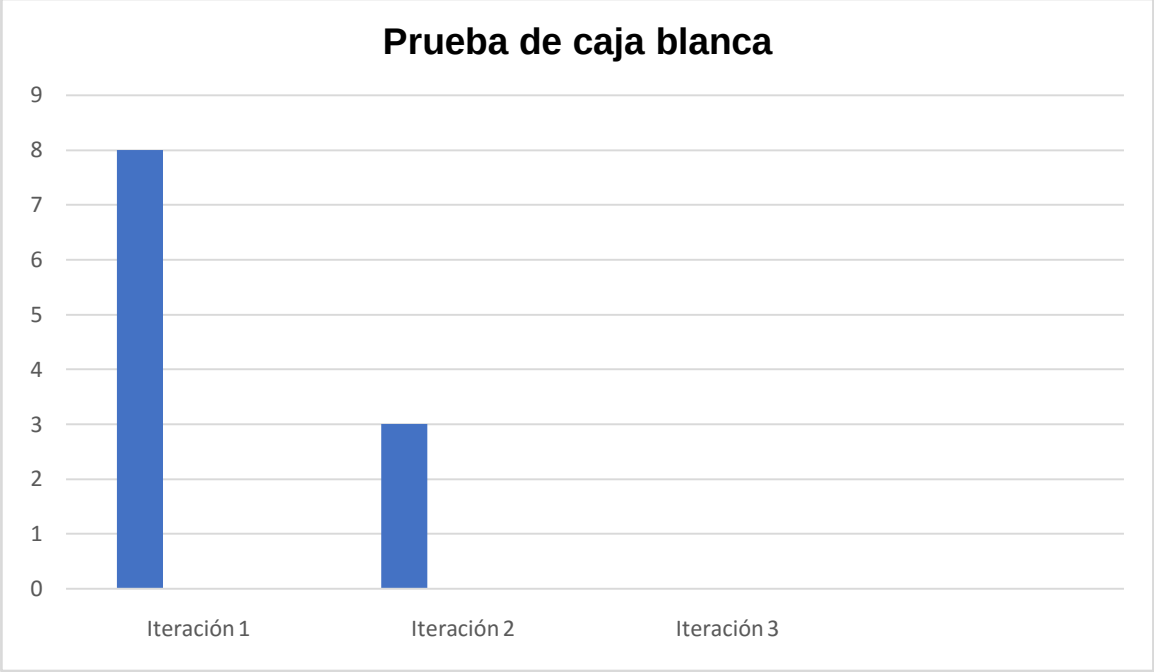

Figura 23. Cantidad de no conformidades por cada iteración prueba de caja blanca

Fuente: Elaboración propia

En la primera iteración realizada se detectaron 8 no conformidades asociadas a errores de validación, entradas y salidas incorrectas. En una segunda iteración se obtuvo 3 no conformidades asociadas a erroresde validación. Posteriormente en una tercera iteración no se obtuvo ninguna no conformidad.

Una vez ejecutados todos los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico para todos los métodos del sistema, se concluye que el código generado está libre de ciclos infinitos y códigos innecesarios. En elanexo 3 se muestra el resto de los casos de pruebas de caja blanca realizados al módulo.

3.5. Pruebas de integración

La prueba deintegración aborda los conflictos asociados con los problemas duales de verificación yconstrucción de programas. Durante la integración, se usan más las técnicas de diseño de casos deprueba que se enfocan en entradas y salidas, aunque también pueden usarse técnicas que ejercitan rutas de programa específicas para asegurar la cobertura de las principales rutas de control. Las pruebas de integración son una técnica sistemática para construir la arquitectura delsoftware mientras se llevan a

cabo pruebas para descubrir errores asociados con la interfaz. Tienen como objetivo identificar errores introducidos por la combinación de programas o componentes probados unitariamente, para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente. [25].

Existe dos tendencias a intentar la integración, incremental y no incremental. La no incremental combina todos los componentes por adelantado. Todo el programa se prueba como un todo. Esto puede provocar un resultado catastrófico debido a q se descubre un conjunto de errores. La corrección se dificulta pues el aislamiento de las causas se complica por la vasta extensión de todo el programa. Una vez corregidos estos errores,otros nuevos aparecen y el proceso continúa en un bucle aparentemente interminable. La integración incremental construye el programa yprueba en pequeños incrementos, donde los errores son más fáciles de aislar y corregir; las interfaces tienen más posibilidades de probarse por completo; ypuede aplicarse un enfoque deprueba sistemático [25]. Es por ello que en la presente investigación se utilizará la integración incremental para comprobar la calidad del módulo.

Integración incremental: Comienza la construcción y la prueba con componentes inferiores dentro de la estructura del programa.Puesto que los componentes se integran deabajo hacia arriba, la funcionalidad que proporcionan los componentes subordinados en determinado nivel siempre está disponible y se elimina la necesidad de representantes, de esta forma los errores son más fáciles de aislar y corregir [25].

Al finalizar la prueba de integración no fueron detectados errores asociados a la interacción del módulo Selectividad con el módulo Estudio API que integra el subsistema Control de Personas, como se muestra en la siguiente tabla:

Tabla 7. Prueba de Integración-módulo Estudio API

Módulo al que se integra	Estudio API
Condiciones de ejecución	El módulo Estudio API ya ha introducido los datos en la base de dato central y exista una conexión con la misma.
Descripción de la prueba	Comprobar que el módulo Selectividad sea capaz de aplicar los perfiles de riesgo a la información gestionada por el módulo Estudio API.
Entradas/Pasos de ejecución	El módulo Estudio API introduce en la base de datos central los datos y cuando se aplican los perfiles de riesgo sobre esta información el módulo Selectividad realiza la selección de las personas que cumplan con dichos perfiles.

Resultado esperado	Se realiza la selección de las personas que cumplan con los perfiles de riesgo aplicados.
Evaluación	Prueba satisfactoria.

Fuente: Elaboración propia

3.6. Pruebas de aceptación

Las pruebas de aceptación son donde los clientes prueban un sistema para decidir si está o no listo para ser aceptado y desplegado en el entorno del cliente. Son una parte inherente del desarrollo de sistemas personalizados. Implican a un cliente que prueba de manera formal un sistema, para decidir si debe o no aceptarlo del desarrollador del sistema [25].

Las pruebas de aceptación deben dirigirse a probar tanto las características funcionales como las no funcionales del sistema (por ejemplo, el rendimiento). Lo ideal sería que dieran cobertura completa a los requisitos del sistema. En la práctica, es difícil establecer criterios de aceptación completamente objetivos. Con frecuencia hay espacio para argumentar sobre si las pruebas deben mostrar o no que un criterio se cubre de manera definitiva, es responsabilidad del cliente verificar la corrección y tomar decisiones acerca de estas pruebas [25].

En la presente investigación se realizará como parte de las pruebas de aceptación la prueba alfa la cual se lleva a cabo en el sitio del desarrollador por un grupo representativo deusuarios finales. El software se usa en un escenario natural con el desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso. Las pruebas alfa serealizan en un ambiente controlado [25].

Tomándose como referencia la especificación de requisitos (ver expediente de Selectividad) y comprobando que el sistema cumple satisfactoriamente los requisitos del cliente, emitiéndose un acta de aceptación (ver anexo 2) garantizándose la evaluación del grado de calidad del software.

Conclusiones del capítulo

La validación de los requisitos permitió una mejor redacción y especificación de los mismos paraelevar la calidad del módulo implementado dándole una mayor satisfacción al cliente. Con la realización de las pruebas de caja negra se validaron los elementos funcionales del módulo y la implementación de cada uno de los requisitos capturados correctamente. Las pruebas de caja blanca validaron la calidad del código, y finalmente con la aplicación de la técnica alfa en las pruebas de aceptación se firmó el acta de aceptación del cliente logrando la satisfacción del mismo.

Conclusiones

Una vez concluida la fundamentación teórica que sustentó la presente investigación, se realizó el análisis, diseño e implementación de módulo y se le realizaron las pruebas establecidas se puede arribar a las siguientes conclusiones:

- La utilización de las herramientas, lenguajes y tecnologías definidas previamente del proyecto GINA permitió establecer la infraestructura tecnológica para la implementación del módulo propuesto.
- Con la aplicación del proceso de desarrollo de software con enfoque ágil basado en el nivel 2 de CMMI, se logró obtener y especificar requisitos, así como la elaboración de los artefactos propuestos y el diseño de la solución.
- Con la implementación del módulo Selectividad se logró incorporar al sistema GINA una de las funcionalidades más importantes que se realizan en la AGR, haciéndolo más completo. Además, se logró la salida de diferentes reportes que permitirán llevar a cabo un control de las actividades que se realizan haciendo uso de interfaces de fácil uso.
- Con la aplicación de la estrategia de prueba defina para la validación del módulo se logró una aplicación que cumple con los requisitos capturados a partir de las necesidades del cliente.
- Se contribuyó con la detección de pasajeros que pudieran ser infractores en la AGR, siendo este un factor decisivo en dicha entidad para la protección de la seguridad nacional, se tributó a un mejor aprovechamiento del tiempo de análisis de los pasajeros, así como la efectividad del trabajo realizado por parte del analista de Aduana, se logró obtener una mejor accesibilidad a la información manejada referente al proceso de selectividad de pasajeros.

Recomendaciones

- Desarrollar un futuro mantenimiento del módulo que permita mantener en óptimas condiciones las funcionalidades del módulo Selectividad.
- Continuar el estudio del negocio para la incorporación de nuevas funcionalidades.

Bibliografía referenciada

1. [1] MSc. Rolando Quintana Aput, “Valoración del impacto social de la informatización de los trámites municipales en Mayabeque”, vol. 19, No. 1 abril-2016, ISSN 1029-3450, RNPS 1843.
2. [2] Cuba, Aduana General de la República. Sitio Web de la Aduana General de la República de Cuba. Disponible en: http://www.aduana.gob.cu/.[Accedido: 3-nov-2019].
3. [3] AIRCOP, el proyecto de comunicaciones aeroportuarias de UNODC, la AMA y la INTERPOL.[En línea]. Disponible en: www.unodc.org/ropan/es/BorderControl/AIRCOP/aircop.html. [Accedido: 4-febrero-2020].
4. [4] Fondo Fiduciario de UNCTAD para las Negociaciones de Facilitación del Comercio y Nota Técnica № 21, *Sistema Aduanero Automatizado SIDUNEA*. [En línea]. Disponible en: http://www.unctad.org/en/docs/wpd181a2_en.pdf. [Accedido: 4-febrero-2020].
5. [5] M. Saravia, instituto tecnológico superior de apatzingán itsa metodologías de desarrollo de software. [En línea]. Disponible en: https://www.academia.edu/6849887/INSTITUTO_TECNOL%C3%93GICO_SUPERIOR_DE_APATZING%C3%81N_ITSA_METODOLOG%C3%8DAS_DE_DESARROLLO_DE_SOFTWARE. [Accedido 4-nov-2019].
6. [6] Tamara Rodríguez Sánchez, “Metodología de desarrollo para la actividad productiva de la UCI”. La Habana 06-mar-2015.
7. [7] Herramientas CASE. Ingeniería del software. Informática Aplicada a la gestión Pública. Universidad de Murcia. [En línea]. Disponible en: https://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html. [Accedido: 9-nov-2019].
8. [8] Ideal Modeling & Diagramming Tool for Agile Team Collaboration.[En línea]. Disponible en: https://www.visual-paradigm.com/. [Accedido: 9-nov-2019].
9. [9] What is Unified Modeling Language (UML).[En línea].Disponible en: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/. [Accedido: 9-nov-2019].
10. [10] SCHMULLER, J. UML en 24 Horas. [En línea]. Disponible en: http://www.scribd.com/doc/38392190/UML-en-24-Horas. [Accedido: 9-nov-2019].

Bibliografía referenciada

11. [11] DR. Jose Luis Lira Turriza, Tecnológico Nacional de México, Instituto Tecnológico superior de Calkiní en el estado de Campeche, Syllabus, INM-407 “Algoritmos y Lenguajes de Programación”. [En línea]. Disponible en: https://www.itescam.edu.mx/portal/asignatura.php?clave_asig=INM-0407&carrera=IIND0405001&id_d=16. [Accedido: 10-nov-2019].
12. [12] PHP, S. O. PHP. [En línea]. Disponible en: http://www.php.net/archive/2007.php. [Accedido: 9-nov-2019].
13. [13] Sencha Ext JS - Comprehensive JavaScript Framework and UI Components.[En línea]. Disponible en: https://www.sencha.com/products/extjs/. [Accedido: 10-nov-2019].
14. [14] FORO 1 SEMANA 2.docx | Entorno de desarrollo integrado | (Lenguaje de programación), *Scribd*. [En línea]. Disponible en: https://es.scribd.com/document/406559013/FORO-1-SEMANA-2-docx. [Accedido: 10-nov-2019].
15. [15] PhpStorm: el IDE rápido e inteligente para programación en PHP de JetBrains. [En línea]. Disponible en: https://www.jetbrains.com/phpstorm/. [Accedido: 10-nov-2019].
16. [16] Definición de Framework de desarrollo (informática). [En línea]. Disponible en: http://www.alegsa.com.ar/Dic/framework.php. [Accedido: 11-nov-2019].
17. [17] “Documentación sobre Symfony”, *symfony.es*. [En línea]. Disponible en: https://symfony.es/documentacion/. [Accedido: 10-dic-2019].
18. [18] What is a Database Management System? - Definition from WhatIs.com, *SearchSQLServer*. [En línea]. Disponible en: https://searchsqlserver.techtarget.com/definition/database-management-system. [Accedido: 10-dic-2019].
19. [19] Oracle Corporation. ORACLE. [En línea]. Disponible en: http://www.oracle.com/us/index.html. [Accedido: 12-dic-2019].
20. [20] del Castillo San Félix, Álvaro. *El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha.*[En línea]. Disponible en: http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html. [Accedido: 2-feb-2020].
21. [21] “Axure software solutions”. [En línea]. Disponible en: https://www.axure.com. [Accedido: 12-feb-2020].
22. [22] J. M. Scoot Chacon Alejandro Fernández, Javier Eguiluz, Aut., *Pro Git, el libro oficial de Git*.
23. [23] François Zaninotto, FabienPotencier. Symfony Guía definitiva, 2008. [En línea]. Disponible en:www.librosweb.es. [Accedido: 12-feb-2020].

24. [24] Sommerville, Ian. 2011. Ingeniería de Software. 9na. México: Addison-Wesley, 2011. ISBN: 978-607-32-0603-7.
25. [25] Pressman, Roger S. Ingeniería de Software. Un enfoque práctico. Séptima, McGraw-Hill INTERAMERICA EDITOR, 2010.
26. [26] O. Blancarte, Introducción a los patrones de diseño - Un enfoque práctico, *Introducción a los patrones de diseño*. [En línea]. Disponible en: https://reactiveprogramming. [Accedido:20-feb-2020].
27. [27] C. Salazar, Alta cohesión y bajo acoplamiento en la programación orientada a objetos | Codesolt.
28. [28] Centro de Informatización de la Gestión de Entidades, Universidad de las Ciencias Informáticas. CEIGE_GINA_1.2_Estándares de codificación para PHP.

Glosario de términos

Flujo:secuencia de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan y cuál es su orden de ejecución.

Proceso: conjunto de actividades relacionadas lógicamente que producen una salida o resultado.

Multiplataforma: término usado para referirse a la característica de aplicaciones de ejecutarse en diversas plataformas o sistemas operativos.

Internet: red de computadoras alrededor de todo el mundo que comparten información unas con otras por medio de páginas o sitios.

Anexos

Anexos 1: Entrevista realizada durante el levantamiento de información.

¿Cuál es la misión de la AGR?

¿Cuál es la visión de la AGR?

¿Qué es la selectividad en la AGR?

¿Quién lleva a cabo este proceso?

¿Qué es un criterio de selección en la AGR?

¿Qué es un operador de riesgo AGR?

¿Qué es un perfil de riesgo en la AGR?

¿Qué es un indicador de riesgo?

¿Cuál es el objetivo de la selectividad?

¿La persona que define los criterios es la misma que aplica los perfiles de riesgo?

¿Qué campos son necesarios para la creación de un perfil de riesgo?

¿Cuáles campos no pueden faltar en la creación de un perfil?

¿Qué sucede si una persona cumple con un perfil de riesgo?

¿Quién es el encargado de aplicar los perfiles de riesgo?

¿A quién se le aplican los perfiles de riesgo?

¿Cuáles son los datos definidos por la AGR?

¿Cuáles son los datos modificables de un perfil de riesgo?

¿Qué se hace cuando un perfil deja de ser efectivo?

¿Cómo se definen los criterios de selección, en qué se basan?

¿Cómo se realiza actualmente el proceso de selectividad?

¿Qué sistemas utilizaba la AGR antes de la implementación de GINA?

¿Qué características tenían estos sistemas?

¿A dónde llega la información adelantada de pasajeros de un vuelo?

¿Qué sistema estuvo utilizando la AGR anteriormente?

¿Qué características tenían estos sistemas?

¿Qué funciones realizaban?

¿Por qué la necesidad de implementar un nuevo sistema?

Anexos 2: Carta de aceptación del cliente.

Anexos 3: Diagramas de Procesos de Negocio.

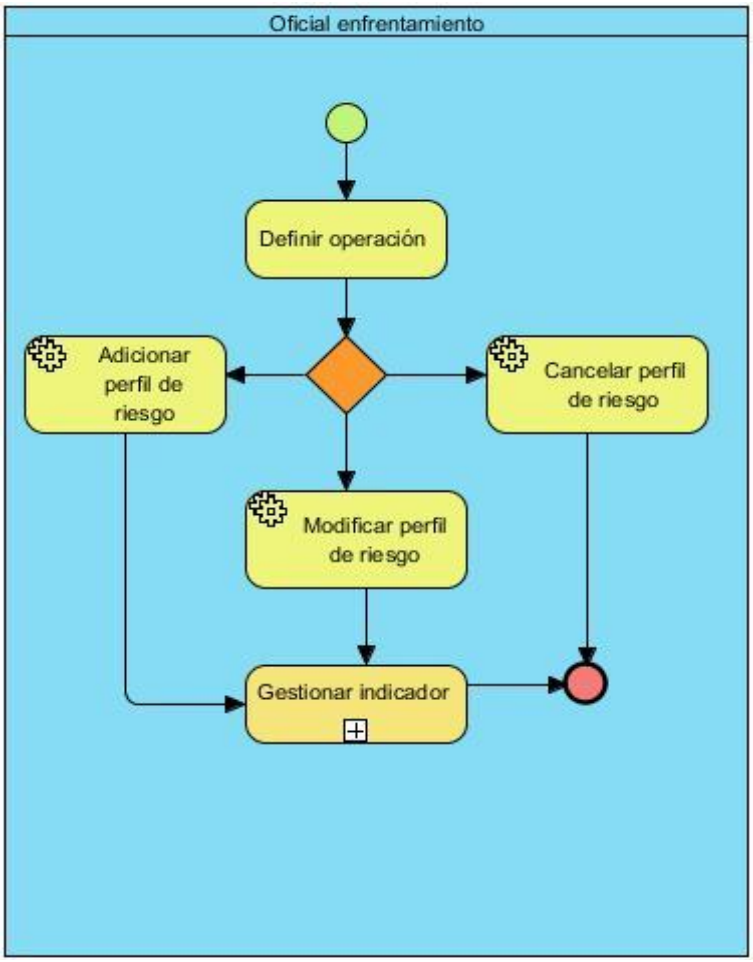

Figura 24. DPN Gestionar perfiles de riesgo

Fuente: Elaboración propia

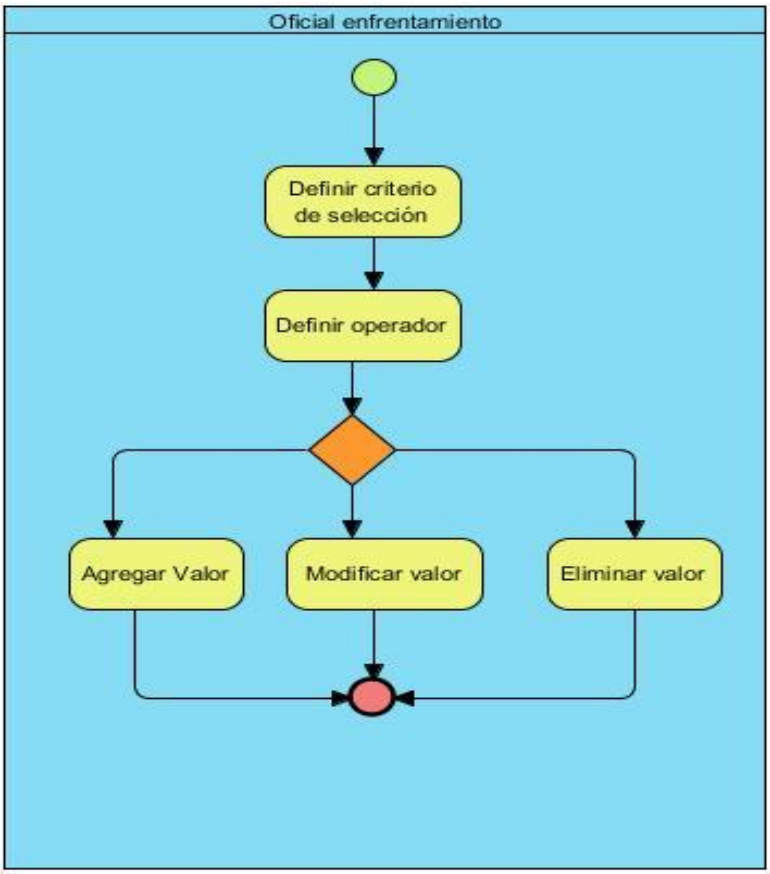

Figura 25. DPN Gestionar indicador

Fuente: Elaboración propia

Anexos 4: Diseño de caso de prueba.

Tabla 8. Diseño de caso de prueba. Introducir perfil de riesgo

Escenarios	Descripción	Fecha inicio	Descripción	Modalidad	Línea de enfrentamiento	Indicador	Medidas a tomar	Respuesta del sistema	Flujo central
EC 1.1 Insertar datos correctamente.	Mediante este escenario se introducen los datos necesarios para insertar un perfil de riesgo.	V 12/06/2020	V Fecha de vencimiento de documentos.	V Evasión del control aduanal.	V Contrabando.	V Fecha de vencimiento del documento.	V Arco en C.	El sistema adiciona el perfil de riesgo correctamente.	El usuario una vez autenticado en el sistema selecciona el subsistema Control de Personas. El sistema muestra las opciones del menú principal y el usuario selecciona la opción Estudio API, gestionar perfiles (EA). De esta interfaz selecciona la opción Adicionar.
EC 1.2 No insertar valores en los campos.	Mediante este escenario no se introduce los valores correspondientes a los campos	I Vacío	I Vacío	I Vacío	I Vacío	I Vacío	I Vacío	El sistema muestra encima del campo una línea de color rojo indicando que este	El usuario una vez autenticado en el sistema selecciona el subsistema Control de

	fecha inicio, descripción, línea de enfrentamiento, modalidad, indicador y medidas a tomar.							campo es obligatorio.	Personas. El sistema muestra las opciones del menú principal y el usuario selecciona la opción Estudio API, gestionar perfiles (EA). De esta interfaz selecciona la opción Adicionar.
EC 1.3 Insertar incorrectamente la descripción.	Mediante este escenario se introduce incorrectamente la descripción del perfil de riesgo.	V 12/06/2020	I #@><:()ju23	V Evasión del control aduanal.	V Contrabando.	V Fecha de vencimiento del documento.	V Arco en C.	El sistema muestra un mensaje de error diciendo que el campo descripción debe comenzar con letra inicial mayúscula y no debe contener caracteres extraños.	El usuario una vez autenticado en el sistema selecciona el subsistema Control de Personas. El sistema muestra las opciones del menú principal y el usuario selecciona la opción Estudio API, gestionar perfiles (EA). De esta interfaz selecciona la

									opción Adicionar.
EC 1.4 Cancelar operación	Mediante este escenario se cancela la creación del perfil de riesgo.	N/A	N/A	N/A	N/A	N/A	N/A	El sistema muestra un mensaje indicando si estás seguro de cancelar la operación.	El usuario una vez autenticado en el sistema selecciona el subsistema Control de Personas. El sistema muestra las opciones del menú principal y el usuario selecciona la opción Estudio API, gestionar perfiles (EA). De esta interfaz selecciona la opción Adicionar.

Fuente: Elaboración propia

Anexos 5: Casos de prueba de caja blanca

Tabla 9. Caso de prueba para el camino básico #1

Caso de pruebas: Camino básico # 1	
Entrada	Se introduce la variable *$idPerfilRiesgo* por la cual se va a realizar la búsqueda
Resultados esperados	Se retorna la variable *$result*como un objeto
Condiciones	El método que se ejecuta en la consulta es un *doSelectOne*

Fuente: Elaboración propia

Tabla 10. Caso de prueba para el camino básico #2

Caso de pruebas: Camino básico # 2	
Entrada	Se introduce la variable *$idPerfilRiesgo* por la cual se va a realizar la búsqueda
Resultados esperados	Se retorna la variable *$result*como un objeto
Condiciones	El método que se ejecuta en la consulta es un *doSelectOne*

Fuente: Elaboración propia

Tabla 11. Caso de prueba para el camino básico #3

Caso de pruebas: Camino básico # 3	
Entrada	Se introduce la variable *$idPerfilRiesgo* por la cual se va a realizar la búsqueda
Resultados esperados	Se retorna la variable *$result*como un objeto
Condiciones	El método que se ejecuta en la consulta es un *doSelectOne*

Fuente: Elaboración propia

Tabla 12. Caso de prueba para el camino básico #4

Caso de pruebas: Camino básico # 4	
Entrada	Se introduce la variable *$idPerfilRiesgo* por la cual se va a realizar la búsqueda
Resultados esperados	Se retorna la variable *$result*como un arreglo
Condiciones	El método que se ejecuta en la consulta es un *doSelect*

Fuente: Elaboración propia

Tabla 13. Caso de prueba para el camino básico #5

Caso de pruebas: Camino básico # 5	
Entrada	Se introduce la variable *$idPerfilRiesgo* por la cual se va a realizar la búsqueda
Resultados esperados	Se retorna la variable *$result*como un arreglo
Condiciones	El método que se ejecuta en la consulta es un *doSelectOne*

Fuente: Elaboración propia