Universidad de las Ciencias Informáticas

Facultad 1

Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de la Aeronáutica Civil de Cuba versión 2.0

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

Autor

Yunier Pérez Bejerano

Tutores

MSc. Damaris Cruz Amarán

Ing. Osay González Fuentes

Ing. Alexei Alayo Rondón

La Habana, junio de 2019
"Año 61 de la Revolución"

"La Revolución empieza ahora, la Revolución no será una tarea fácil, la Revolución será una empresa dura y llena de peligros."

Fidel Castro Ruz

Dedicatoria

Dedico el presente trabajo de diploma a mi madre que con su demonstración de una madre ejemplar me ha enseñado a no desfallecer ni rendirme ante nada y siempre perseverar a través de sus sabios consejos.

A mi novia Beatriz y mi hija que está por nacer en este momento crucial de mi vida.

A mi papa por siempre estar ahí para mí.

A todos mis familiares, amigos y profesores que creyeron en mí y en este trabajo de diploma que hoy se hace realidad.

A todas las personas que hoy están aquí conmigo que también son protagonistas de este triunfo.

Agradecimientos

A mi madre que con su empeño y dedicación ha sido faro y guía en todos los momentos de mi vida, por su apoyo incondicional y estar siempre pendiente de mí y de mis estudios, hoy está aquí celebrando este triunfo.

A mi novia Beatriz por tener tanta paciencia conmigo, por estar a mi lado, ayudarme siempre y dedicarme todo su amor y ternura. También a toda su familia que siempre me han ayudado y apoyado.

A mi papa Sergio por enseñarme los valores que hoy conforman mi persona.

A leo que ha sido como un padre para mí y siempre me ha brindado todo su apoyo.

A mi tutora Damaris por todo su apoyo, comprensión y paciencia durante todo el tiempo que estuve inmerso en este largo proceso.

Agradezco a la rectoría de la universidad (por la posibilidad de la continuidad del plan de estudios) por permitir graduarme, así como se han graduado más de 14 mil ingenieros, logrando un impacto considerable en la informatización del país.

A la dirección del Centro de Identificación y Seguridad Digital especialmente a mis tutores Osay y Alexei que estuvieron presente en todo momento que los necesite.

A toda mi familia en general por la confianza que siempre me han tenido y por el apoyo que me han dado.

A la valiosa colaboración de la facultad 1 en mi formación como profesional.

A todo el claustro de profesores por la participación activa, abnegada y desinteresada teniendo dedicación al enriquecimiento de mis conocimientos y experiencias que aparecen en este trabajo de diploma.

A mi amigo Elvis que siempre me ha brindado su apoyo.

A todas mis amistades de mi antiguo grupo y grupo actual, a Elvis y Yosbel, Javier, entre otros.

A la revolución cubana y al comandante Fidel Castro por ser el creador de la Universidad de las Ciencias Informáticas.

Declaración jurada de autoría

Declaro por este medio que yo Yunier Pérez Bejerano, con carné de identidad 93030619781 soy el autor principal del trabajo titulado “Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de Aeronáutica Civil de Cuba versión 2.0” y autorizo a 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 firmo la presente declaración jurada de autoría en La Habana a los _____ días del mes de _____ del año _____.

_____

Firma del autor

Yunier Pérez Bejerano

_____

Firma del tutor

Ing. Osay González Fuentes

_____

Firma del tutor

Ing. Alexei Alayo Rondón

_____

Firma de la tutora

MSc. Damaris Cruz Amarán

Resumen

El Instituto de la Aeronáutica Civil de Cuba (IACC) cuenta con una aplicación informática desarrollada en la Universidad de las Ciencias Informáticas (UCI) que permite llevar a cabo los procesos de gestión y emisión de licencias que se realizan en la institución. La aplicación realiza el registro de titulares en el sistema, el otorgamiento de licencias a titulares, la renovación de licencias y facilita la generación e impresión de licencias y certificados de titulares. También, brinda la posibilidad de personalizar los documentos de identificación de los titulares y la gestión de las habilitaciones. A su vez, provee la generación de un conjunto de notificaciones y reportes necesarios para los usuarios del sistema, agilizándose el acceso a la información deseada. Esta aplicación lleva dos años en explotación y cuenta con soporte por parte del Centro de Soporte de la UCI. Sin embargo, en la actualidad, el sistema presenta problemas de seguridad, en su rendimiento y existe la necesidad, por parte del cliente, de generar nuevos reportes para mejorar y asistir la toma de decisiones. El presente trabajo expone como solución el desarrollo e implementación de la versión 2.0 del Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de Aeronáutica Civil de Cuba. El trabajo estuvo guiado por la metodología AUP-UCI, la cual garantizó un desarrollo ágil y organizado.

Palabras clave: Gestión y emisión de licencia aeronáutica, Instituto de la Aeronáutica Civil de Cuba, Universidad de las Ciencias Informáticas.

Índice de contenidos

INTRODUCCIÓN

Introducción

Entre la variedad de tipos de sistemas existentes sustentados en las Tecnologías de la Información y las Comunicaciones (TIC), se encuentran los de gestión de información. De forma específica, los Sistemas de Gestión de Licencias Aeronáuticos se diseñan para otorgar, renovar, actualizar y convalidar las licencias y habilitaciones del personal aeronáutico civil de conformidad con las regulaciones técnicas de aviación civil nacionales e internacionales. (Instituto de Aeronáutica Civil de Cuba, 2017)

Cada país, cuenta con una organización para dirigir y gestionar la actividad de la aviación civil. En el caso de Cuba se denomina, Instituto de la Aeronáutica Civil (IACC). Esta institución, tiene la misión de ejecutar las funciones relacionadas con el ejercicio de la Autoridad Aeronáutica en el territorio Nacional. Para una mejor gestión de su labor fueron informatizados sus procesos. La Universidad de las Ciencias Informáticas (UCI), a través del Centro Identificación y Seguridad Digital (CISED) adscrito a la Facultad 1 fue la entidad encargada de la informatización.

CISED desarrolla productos, servicios y soluciones integrales en el campo de la identificación y la seguridad digital, con una alta confiabilidad, precisión y eficiencia económica, con personal altamente competente y calificado. Esto le permite adaptarse a las necesidades de los clientes y orientarlos hacia los últimos avances de la tecnología, garantizando la formación y superación de los estudiantes y especialistas de las diferentes áreas, altamente comprometidos con la Revolución, los clientes y la organización. (Universidad de las Ciencias Informáticas, 2018)

El Sistema de Gestión de Licencias del Personal Aeronáutico versión 1.0 es un sistema desarrollado por CISED para el IACC. Es un sistema integrado que informatiza dos procesos importantes: la gestión y emisión de licencias. Está diseñado para otorgar, renovar, actualizar y convalidar las licencias y habilitaciones del personal técnico aeronáutico civil. Su explotación alcanza los dos años de vida y presenta en la actualidad la siguiente situación:

1. Las tecnologías del Sistema de gestión de licencias aeronáuticas están obsoletas. La tecnología obsoleta trae como consecuencia la afectación del funcionamiento del departamento de licencias retrasando el trabajo del personal y también que el sistema presente problemas de seguridad debido a que no se ha actualizado desde su puesta en marcha esto trae como consecuencia que los datos de la empresa no estén seguros.

INTRODUCCIÓN

2. Vulnerabilidades en los procesos de emisión y gestión de licencias a partir de problemas de seguridad en dos paquetes del núcleo de *Symfony* (*Symfony/polyfill-php55* v1.4.0 y *Symfony/Symfony* v2.8.22) y cada uno de estos paquetes en varios de sus componentes. Estos resultados se obtuvieron del análisis de seguridad de la versión 1.0 del Sistema de Gestión de Licencias del Personal Aeronáutico.
3. El sistema de gestión de licencias aeronáuticas cuenta con tres reportes para la toma de decisiones. Los directivos de la entidad, a partir de su experiencia en la explotación de la aplicación, tienen necesidad de nuevos y mejores reportes para un mayor control de los procesos que se realizan y mejorar la toma de decisiones informada. Su valoración, es que a través de los reportes actuales no se obtiene la información necesaria para su gestión organizacional.
4. El rendimiento de la base de datos es bajo lo cual provoca que la gestión digital de los procesos de negocio dentro de la organización vea afectada su capacidad de respuesta. Esto ha conllevado a repercusiones en la entrega final de la documentación en cuánto a tiempo de espera, lo cual afecta la calidad del servicio y produce insatisfacciones en los usuarios finales. De igual forma, existen demoras en las solicitudes durante la realización de las consultas y se extiende el tiempo de espera del personal aeronáutico que trabaja en el departamento de licencias.

A partir de la situación problemática planteada se define como **problema de la investigación:** ¿Cómo mejorar los procesos de gestión y emisión de licencias del personal aeronáutico del Instituto de la Aeronáutica Civil de Cuba?

El **objeto de estudio** se determina como el proceso de gestión y emisión de licencias aeronáuticas, y como **campo de acción** el proceso de gestión y emisión de licencias aeronáuticas del personal aeronáutico del Instituto de la Aeronáutica Civil de Cuba en el Sistema de Gestión de Licencias del Personal Aeronáutico versión 1.0.

Para dar cauce a la investigación se declara como **objetivo general** desarrollar la versión 2.0 del Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de la Aeronáutica Civil de Cuba para la mejora de los procesos de gestión y emisión de licencias del personal aeronáutico.

El objetivo general se desglosa en los **objetivos específicos**:

INTRODUCCIÓN

- ✓ Elaborar el marco teórico-metodológico de la investigación para el desarrollo de la versión 2.0 del Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de la Aeronáutica Civil de Cuba.
- ✓ Determinar las metodologías, tecnologías y herramientas que conforman el ambiente de desarrollo para el diseño del Sistema de Gestión de Licencias del Personal Aeronáutico versión 2.0.
- ✓ Realizar el análisis y diseño de las funcionalidades para el desarrollo de los procesos de gestión y emisión en el Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de la Aeronáutica Civil de Cuba en su versión 2.0.
- ✓ Implementar los artefactos de la versión 2.0 para el desarrollo de los procesos de gestión y emisión en el Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de la Aeronáutica Civil de Cuba.
- ✓ Validar la versión 2.0 del Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de la Aeronáutica Civil de Cuba.

Para guiar la investigación se plantean las siguientes **preguntas científicas**:

- ✓ ¿Cuáles son los principales referentes teórico-metodológicos que sustentan la investigación para el desarrollo del Sistema de Gestión de Licencias del Personal Aeronáutico versión 2.0?
- ✓ ¿Cuáles metodologías, tecnologías y herramientas, conforman el ambiente de desarrollo para la implementación del Sistema de Gestión de Licencias del Personal Aeronáutico versión 2.0?
- ✓ ¿Cuáles elementos deben tenerse en cuenta en el análisis y diseño de la propuesta de solución?
- ✓ ¿Cómo implementar los artefactos a partir del análisis y diseño del Sistema de Gestión de Licencias del Personal Aeronáutico 2.0?
- ✓ ¿Cómo validar las funcionalidades de la solución propuesta para la mejora de los procesos de gestión y emisión de licencias del personal aeronáutico?

Para el desarrollo de la investigación se emplearon los siguientes **métodos científicos**:

Métodos teóricos:

INTRODUCCIÓN

- ✓ **Histórico-Lógico:** Se utiliza para estudiar la evolución y el comportamiento de soluciones similares a la propuesta de solución. También para conocer los antecedentes y tendencias actuales de los sistemas de gestión de licencias aeronáuticas.
- ✓ **Analítico-Sintético:** Mediante este método se identifican conceptos y definiciones importantes relacionadas con el tema, permitiendo generar una propuesta adecuada a la situación planteada.
- ✓ **Análisis documental:** Para la revisión bibliográfica y la revisión de las fuentes primarias de investigación.
- ✓ **Modelación:** Para la modelación de las funcionalidades del flujo de la aplicación. Permitió la abstracción de la realidad al modelo informático y de ahí buscar la manera de implementarlo, concretando las características de cada uno de los componentes seleccionados.

Métodos empíricos:

- ✓ **Entrevista:** Se utiliza para la captura de requisitos; en la que se realiza una conversación planificada entre el investigador y los funcionarios para la obtención de los requisitos. Esto constituye uno de los mayores afluentes del conocimiento para la descripción de los requisitos funcionales del Sistema de Gestión de Licencias del IACC 2.0.
- ✓ **Observación:** Este método es utilizado para observar las deficiencias existentes en el funcionamiento del Sistema de Gestión de Licencias del IACC 1.0, también investigar sobre el funcionamiento, ventajas y desventajas de otros Sistemas de Gestión de Licencias con características similares.

El presente documento está estructurado en tres capítulos

Capítulo 1: Plantea la fundamentación teórica de la investigación, e incluye un estudio del estado del arte del tema que se investiga. A su vez, se describen las tecnologías y herramientas que se utilizan en la solución desarrollada.

Capítulo 2: Describe el proceso de análisis y diseño, el cual comienza con el planteamiento y planificación de las historias de usuario que describen las funcionalidades a desarrollar. Detalla la arquitectura de la solución y sus principales características.

Capítulo 3: Describe las fases de implementación y pruebas y los componentes necesarios utilizados en la solución. Se implementan todas las funcionalidades identificadas, logrando un sistema que satisface las

INTRODUCCIÓN

principales necesidades del cliente. Se detallan también las pruebas que se le realizaron al sistema, con el objetivo de asegurar la calidad y eficiencia de la solución y comprobar su validez.

Capítulo 1. Fundamentación Teórica de la investigación

En este capítulo se lleva a cabo toda la primera fase de la metodología de desarrollo, se establecen conceptos fundamentales para la comprensión de la investigación. Se realiza un análisis del estado del arte de Sistemas de Gestión de Licencias existentes y que posean características similares o que tengan alguna función semejante a la del sistema en estudio. Se realiza, además un estudio de las herramientas y tecnologías utilizadas para dar solución al problema de investigación.

1.1 Conceptos y definiciones asociados al dominio del problema

Se hace necesaria la exposición de conceptos relacionados con la aeronáutica civil. Cada uno de ellos, se relacionan a continuación a partir de lo estipulado por (Instituto de Aeronáutica Civil de Cuba, 2017)

"**Aeronave:** Toda máquina que puede sustentarse en la atmósfera por reacciones del aire que no sean las reacciones del mismo contra la superficie de la tierra" (p.1.63-A-1).

"**Autoridad Aeronáutica:** En materia de aviación civil la Autoridad Aeronáutica la ostenta y ejerce el Ministerio del Transporte a través del Instituto de Aeronáutica Civil de Cuba" (p.1.63-A-1).

"**Autoridad Otorgadora de Licencias (AOL):** El Departamento de Licencias e Instrucción del Instituto de Aeronáutica Civil de Cuba" (p.1.63-A-1).

Se considera a la Autoridad Otorgadora de Licencias como encargada de:

- ✓ Evaluar la idoneidad del candidato para ser titular de una licencia o habilitación.
- ✓ Expedir y anotar licencias y habilitaciones.
- ✓ Designar y autorizar a las personas aprobadas.
- ✓ Aprobar los cursos de instrucción.
- ✓ Centralizar las actividades relativas a la aprobación del uso de dispositivos de instrucción para simulación de vuelo y autorización para dicho uso con objeto de adquirir la experiencia o demostrar la pericia exigida para la expedición de una licencia o habilitación.
- ✓ Convalidar las licencias expedidas por otros estados contratantes.

- ✓ Centralizar las actividades relativas a la certificación o aceptación y la vigilancia de los centros de instrucción y de entrenamiento de aeronáutica civil. (p.1.63-A-1).

"**Avión (aeroplano):** Aerodino propulsado por motor, más pesado que el aire, que debe su sustentación en vuelo principalmente a reacciones aerodinámicas ejercidas sobre superficies que permanecen fijas en determinadas condiciones de vuelo" (p.1.63-A-1).

"**Convalidación (de una licencia):** El acto por el cual el IACC, en vez de otorgar su propia licencia, reconoce como equivalente a la suya propia la otorgada por otro Estado contratante" (p.1.63-A-1).

Conversión (de una licencia): Método por el cual el IACC otorga una licencia nacional basándose en una licencia extranjera, válida y vigente, emitida por un Estado contratante al Convenio sobre Aviación Civil Internacional, a una persona nacional o extranjera, previo cumplimiento de los requisitos establecidos para tal fin. (p.1.63-A-1)

Simulador de vuelo: Proporciona una representación exacta del puesto de pilotaje de un tipo particular de aeronave, hasta el punto de que simula positivamente las funciones de los mandos de las instalaciones y sistemas mecánicos, eléctricos, electrónicos, etc., de a bordo, el medio ambiental normal de los miembros de la tripulación de vuelo, y la performance y las características de vuelo de ese tipo de aeronave. (p.1.63-A-2)

"**Habilitación:** Autorización inscrita en una licencia de personal aeronáutico o asociado con ella, y de la cual forma parte, en la que se especifican condiciones especiales, atribuciones o restricciones referentes a dicha licencia" (p.1.63-A-2).

Instrucción inicial: Es la etapa primaria de instrucción, donde se proporcionan los conocimientos teóricos y prácticos pertinentes para el desempeño de una ocupación o cargo. Para los portadores de licencias aeronáuticas, se habilita la misma, culminado el período de instrucción establecido. (p.1.63-A-2)

Instrucción periódica: Es la etapa de instrucción para el personal ya certificado o habilitado para el puesto que desempeña, donde recibe la preparación teórico-práctica que se requiere para la continuación del desempeño de sus funciones. Esta instrucción puede incluir temáticas para la superación, el refrescamiento o la actualización relacionados con nuevas amenazas, cambios en las

tecnologías, así como el conocimiento de las políticas, directrices, regulaciones, procedimientos y procesos que hayan sufrido modificaciones. (p.1.63-A-2)

"**Libro Oficial de Vuelo (LOV):** Libro personal de registro de vuelo, en que se consigna en forma cronológica el tiempo de vuelo de un titular de licencia" (p.1.63-A-2).

"**Licencia:** Documento oficial otorgado por la AOL, que indica la especialidad aeronáutica del titular y las restricciones en caso de haberlas, y le otorga la facultad para desempeñar las funciones propias de las habilitaciones expresamente consignadas en ella" (p.1.63-A-2).

"**Renovación:** Acto administrativo por el cual al titular de una licencia se le restablece la o las atribuciones que la misma le confiere, una vez cumplidos los requisitos establecidos en la presente regulación" (p.1.63-A-3).

1.2 Sistema de gestión de Licencias IACC 1.0

El Sistema de Gestión de Licencias al personal Aeronáutico del Instituto de Aviación permite llevar a cabo los procesos que se realizan actualmente en el departamento de licencias del IACC. (Instituto de Aeronáutica Civil de Cuba, 2017)

Este sistema permite:

- ✓ Renovar, prorrogar y convalidar licencias.
- ✓ El registro de titulares en el sistema.
- ✓ La generación e impresión de licencias y certificados de titulares.
- ✓ Personalizar los documentos de identificación de los titulares y la gestión de sus habilitaciones.
- ✓ La generación de un conjunto de notificaciones y reportes necesarios para los usuarios del sistema, agilizándose el acceso a la información deseada.
- ✓ Gestión de usuarios del sistema.
- ✓ Gestión de titulares internos y externos.
- ✓ Gestión de miembros de tripulación.
- ✓ La inicialización de los tipos de licencias titular, titular externo y miembro de tripulación.

- ✓ Gestión de centro y programa de instrucción.
- ✓ Gestión de los simuladores de vuelo.

En este sistema después de analizado se detectó que presenta problemas de seguridad, su rendimiento es bajo, las tecnologías están obsoletas y existe la necesidad de nuevos reportes para la toma de decisiones.

1.3 Análisis de sistemas homólogos

En el análisis de la bibliografía realizada se tienen en cuenta para el estudio de la presente investigación los siguientes sistemas. A continuación, se exponen los resultados obtenidos de su estudio:

1.3.1 Sistema de Información de Gestión Aeronáutica (SIGA)

SIGA es el sistema de información de gestión aeronáutica de colombia. Es utilizado para realizar exámenes teóricos y tramitar ante del grupo de licencias técnicas y de exámenes, cualquier proceso de expedición, adición o convalidación de licencias. Permite el entrenamiento, la realización de exámenes teóricos, y tramitar ante el grupo de licencias técnicas y de examen, cualquier examen de procesos de expedición, adición o convalidación de licencias. Permite a los centros de instrucción registrar directamente en el aplicativo la información de sus alumnos y egresados. (Aerocivil, 2018)

1.3.2 Sistema Informático de Personal Aeronáutico (SIPA)

SIPA es un sistema integrado y diseñado para administrar el ciclo de vida de las licencias y habilitaciones del personal aeronáutico civil de Chile. Este sistema permite ingresar en línea las solicitudes de licencias aeronáuticas. La Dirección General de Aeronáutica Civil (DGAC) dispone de toda la funcionalidad necesaria en el sistema SIPA para procesar las solicitudes, asegurar el cumplimiento de la normativa y recibir toda la información requerida desde otras organizaciones participantes en el proceso de licenciamiento. También permite entre sus características:

- ✓ Otorgar, convalidar, revalidar y emitir certificados e impresión de licencias.
- ✓ Impresión de licencia aeronáutica.
- ✓ Enviar, monitorear y tramitar las solicitudes de licencias y/o habilitaciones.
- ✓ Rendir exámenes teóricos en la sesión personal del usuario.
- ✓ Plataforma de exámenes de idioma inglés.

- ✓ Hoja de vida completa y solicitud de información por fechas.
- ✓ Bitácora de vuelo Electrónica.
- ✓ Consultas de Licencias y trazabilidad de información.
- ✓ Pago de Tasas en Línea de Licencias y Comisiones de Exámenes Prácticos. (DGAC, 2019)

1.3.3 Sistema Integrado de Trámites de Gestión Aeronáutica (SITGA)

SITGA es el Sistema Integrado de Trámites de Gestión Aeronáuticos del Instituto Nacional de Aeronáutica Civil de Venezuela (INAC). EL INAC brinda información de los trámites que se realizan en SITGA los cuales son trámites de aeronaves, empresas, personal aeronáutico. El SITGA permite la emisión de licencias aeronáuticas, el otorgamiento de habilitaciones, renovación de licencias aeronáuticas y/o habilitaciones, certificación de licencias y certificados médicos aeronáuticos y la convalidación o reconocimiento de licencias o habilitaciones otorgadas por otro estado contratante. (INAC, 2019)

Resultados del estudio de sistemas homólogos

El estudio realizado permite determinar que existen varios sistemas de gestión de licencias a nivel internacional. Los sistemas presentan similitudes en el proceso de gestión y emisión de licencias a partir de que todos se rigen por las normas establecidas por la Organización de Aviación Civil Internacional (OACI). Permiten las funcionalidades de crear, renovar y convalidar licencias y la generación de la certificación necesaria asociada al proceso.

No obstante, el estudio no pudo realizarse a profundidad. Los sistemas internacionales estudiados no permiten un acceso al módulo que gestiona los procesos de gestión y emisión de licencias que desarrollan. Esto se justifica en la seguridad que poseen como requisito establecido contra personas ajenas a la organización. Cada organización teniendo en cuenta ello, debe desarrollar sus propios sistemas. El estudio sirve para conocer las similitudes declaradas en cuanto a funcionalidades, las cuales son descritas en el epígrafe.

1.4 Metodologías de desarrollo de software

Las metodologías de desarrollo de software son indispensables para crear o actualizar software de calidad que cumpla con los requisitos de los usuarios; son una parte fundamental de la Ingeniería de software la cual denomina metodología a un conjunto de métodos coherentes y relacionados por unos principios

comunes. Es importante conocer la definición de metodología y desarrollo. Metodología es una palabra compuesta por tres vocablos griegos: metá (“más allá”), odós (“camino”) y logos (“estudio”); considerando lo anterior, la definición de metodología son los métodos para luego determinar cuál es el más adecuado. Lograr la construcción de un sistema informático eficiente, que cumpla con los requisitos planteados, es una tarea realmente intensa y sobre todo difícil de cumplir. Las metodologías para el desarrollo del software imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Las metodologías de desarrollo se pueden dividir en dos grupos de acuerdo con sus características y los objetivos que persiguen: ágiles y robustas. (Tecnología e Innovación, 2015)

Metodología de desarrollo de software a utilizar

La metodología de desarrollo de software a utilizar en la presente investigación es la variación de AUP para la UCI (AUP-UCI) en su escenario 4 debido a que logra estandarizar el proceso de desarrollo de software. Esta es la metodología a utilizar en la línea de productos elaborados en la Universidad de las Ciencias Informáticas.

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición), AUP-UCI mantiene la fase de Inicio, pero modifica el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, la fase de Ejecución y se agrega la fase de Cierre de la siguiente manera:

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 del 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.

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.

Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto. (UCI, 2018)

1.5 Herramientas y tecnologías utilizadas en la propuesta de solución

En la realización de un proyecto es imprescindible la etapa donde se definen las tecnologías y herramientas a utilizar, así como la versión de cada una de ellas que será empleada. Las tecnologías que se emplean para desarrollar la propuesta de solución se seleccionan por las ventajas aprovechables para el desarrollo del sistema de gestión de licencia del IACC.

1.5.1 *Framework PHP Symfony* v3.4.28

Definición: *Symfony* es un *framework* PHP de tipo *full-stack* construido con varios componentes independientes creados por el proyecto *Symfony*. (Symfony, ¿Qué es Symfony?, 2019)

Principales características:

- ✓ Su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia MIT de **software libre**.
- ✓ La **documentación** del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos.
- ✓ Aprender a programar con *Symfony* te permite acceder a una gran variedad de proyectos: el *framework Symfony* 3 para crear aplicaciones complejas, para sitios web sencillos y los componentes *Symfony* para otras aplicaciones PHP.
- ✓ Según *GitHub, Symfony* es el proyecto PHP más **activo**, lo que garantiza que nunca te quedarás atrapado en un proyecto sin actividad. Además, el líder del proyecto, Fabien Potencier, es la segunda persona más activa del mundo en *GitHub*.
- ✓ Aunque en su desarrollo participan cientos de programadores de todo el mundo, las decisiones técnicas importantes siempre las toma Fabien Potencier, líder del proyecto. Esto evita el peligro de que surjan *forks* absurdos y la comunidad se fragmente.
- ✓ Los componentes de *Symfony* son tan útiles y están tan probados, que proyectos tan gigantescos como *Drupal* 8 están construidos con ellos.
- ✓ En todo el mundo se celebran varias conferencias dedicadas exclusivamente a *Symfony*. (Symfony, ¿Qué es Symfony?, 2019)

1.5.2 PHP v7.2.9

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. (Group, 2019). Lo anterior conllevó a la selección de PHP como lenguaje de programación, además, de ser el lenguaje en que esta desarrollado *Symfony* 3.4, es la ventaja de ser un lenguaje del lado del servidor, lo que se entiende como un lenguaje transparente al usuario. Al estar embebido o incrustado en el código HTML le proporciona a este el dinamismo y operabilidad deseados. Permite una potente y sencilla interacción con bases de datos mediante *Doctrine*, proporcionado por *Symfony* 3.4 y que se encarga de tratar el acceso a los datos persistentes.

PHP posee características que favorecen en gran medida el desarrollo de la aplicación propuesta, las cuales son:

- ✓ Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- ✓ Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas.
- ✓ El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador.
- ✓ Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad.
- ✓ Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema 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.
- ✓ No requiere definición de tipos de variables, aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- ✓ Permite el manejo de excepciones. (Group, 2019)

1.5.3 Twig v2.0

Twig es un flexible, rápido y seguro motor de plantillas para PHP. Es a la vez, un amigable ambiente para el diseñador y desarrollador apegado a los principios de PHP, añadiendo útil funcionalidad a los entornos de plantillas.

Características

- ✓ **Rápido:** *Twig* compila las plantillas hasta código PHP regular optimizado. El costo general en comparación con código PHP regular se ha reducido al mínimo.
- ✓ **Seguro:** *Twig* tiene un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable. Esto permite utilizar *Twig* como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.
- ✓ **Flexible:** *Twig* es alimentado por flexibles analizadores léxico y sintáctico. Esto permite al desarrollador definir sus propias etiquetas y filtros personalizados, y crear su propio *DSL*. (Symfony, Twig, 2019)

1.5.4 JavaScript v1.8.5

Es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con este lenguaje se pueden crear diferentes efectos e interactuar con usuarios o los diferentes componentes de una página web. Posee varias características, es basado en acciones que poseen menos restricciones y también es multiplataforma. Gran parte de la programación está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros. Es necesario resaltar que hay dos tipos de *JavaScript*: por un lado, está el que se ejecuta en el cliente, este es el *JavaScript* propiamente dicho, aunque técnicamente se denomina Navegador *JavaScript*. Pero también existe un *JavaScript* que se ejecuta en el servidor, y se denomina *LiveWire JavaScript*. (Web, 2007)

1.5.5 CSS 3

CSS 3 (*Cascading Stylesheets*, Hojas de Estilo en Cascada) es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, entre otros. Una vez creados los contenidos, se utiliza el lenguaje CSS 3 para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y

vertical entre elementos, posición de cada elemento dentro de la página. Separa la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo también llamados documentos semánticos. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (Eguiluz, 2018)

1.5.6 PostgreSQL v9.4

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. *PostgreSQL* utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. A continuación, se muestran las características más importantes y soportadas por *PostgreSQL*:

- ✓ Es una base de datos 100% ACID.
- ✓ Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- ✓ Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- ✓ Copias de seguridad en caliente (*Online/hot backups*).
- ✓ *Unicode.*
- ✓ Juegos de caracteres internacionales.
- ✓ Regionalización por columna.
- ✓ *Multi-Version Concurrency Control* (MVCC).
- ✓ Múltiples métodos de autentificación.
- ✓ Acceso encriptado vía SSL.
- ✓ Completa documentación.

- ✓ Licencia BSD.
- ✓ Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit. (PostgresSQL, 2019)

1.5.7 UML

UML son las siglas de "*Unified Modeling Language*" o "Lenguaje Unificado de Modelado". Es un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software (programas informáticos). UML son una serie de normas y estándares que dicen cómo se debe representar algo. (Krall, 2018)

1.5.8 Herramienta para el modelado, *Visual Paradigm* para UML v8.0

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software. Permite modelar todo tipo de diagramas UML, generar código desde diagramas, generar documentación, realizar ingeniería tanto directa como inversa, entre otras funciones. Este software permite crear modelos de gran calidad en poco tiempo y a un menor costo, y presenta una licencia gratis para uso no comercial. Es multiplataforma y posee una interfaz amigable que permite un mejor desenvolvimiento por parte del equipo de desarrollo. (Visual Paradigm, 2018)

1.5.9 Entorno de desarrollo *JetBrains PhpStorm* v2018.2.5

JetBrains PhpStorm es un IDE comercial multiplataforma para PHP creado en la plataforma *IntelliJ IDEA* de *JetBrains. PhpStorm* proporciona un editor para PHP, HTML y *JavaScript* con análisis de código sobre la marcha, prevención de errores y refactorizaciones automatizadas para código PHP y *JavaScript*. Es uno de los entornos de programación más completos de la actualidad, permite editar código no sólo del lenguaje de programación PHP como lo indica su nombre sino también de los mencionados anteriormente. Entre sus características están las siguientes: (JetBrains, 2019)

1. Permite la gestión de proyectos fácilmente.
2. Proporciona un fácil autocompletado de código**.**
3. Soporta el trabajo con PHP 5.5.
4. Sintaxis abreviada.

Algo que destaca en *PhpStorm* es la ejecución del código en la misma interfaz del IDE. Así como también la interpretación y visualización inmediata de código PHP hasta en cinco (5) de los navegadores web más populares.

Conclusiones parciales del capítulo

En el capítulo se definen los conceptos esenciales para la investigación relacionados con las terminologías de la aeronáutica civil y que son empleados como nomencladores en la propuesta de solución.

El análisis de homólogos permitió establecer la existencia de sistemas que poseen módulos de gestión y emisión de licencias para el personal de la aeronáutica civil. De igual forma, el estudio establece que este tipo de sistema por las regulaciones dictadas por la OACI, no permite la entrada a intrusos, ello redujo el alcance en los elementos investigados. Ello conlleva al desarrollo de un sistema específico para cada organización.

El estudio de la versión 1.0 del Sistema de Gestión de Licencias permitió establecer que el sistema presenta problemas de seguridad, su rendimiento es bajo, la tecnología es obsoleta y existe la necesidad de nuevos reportes para la toma de decisiones.

Para la propuesta de solución se determina el uso de AUP-UCI como metodología de desarrollo, *Visual Paradigm* como herramienta para el modelado, PHP como lenguaje de programación de lado del servidor, *PHPStorm* como entorno de desarrollo, *postgreSQL* como gestor de base de datos, *JavaScript* como lenguaje de programación del lado del cliente y *framework php Symfony*.

Capítulo 2. Análisis y diseño de la propuesta de solución

En este capítulo se presentan las fases de planificación y diseño definidas por la metodología AUP variación UCI escenario cuatro (4). Se realiza la propuesta de solución y se planifica el proceso de desarrollo de software, identificando los requisitos funcionales y no funcionales de la propuesta de solución, patrones de diseño, estilo arquitectónico y se realiza el diseño de las clases. De esta forma se describe la propuesta de solución para el Sistema de Gestión de Licencias del IACC versión 2.0.

2.1 Propuesta de solución

Para dar cumplimento al objetivo planteado se propone desarrollar una segunda versión del Sistema de Gestión de Licencias del Personal Aeronáutico del IACC de tal manera que permita una gestión eficaz de los procesos de gestión y emisión de licencias. Este sistema permite llevar a cabo la gestión de los procesos que se realizan actualmente en la Oficina de Licencias al Personal Aeronáutico del Instituto de Aeronáutica Civil de Cuba. Este sistema 1) permite el registro de titulares en el sistema, el otorgamiento de licencias a titulares y su renovación, 2) facilita la generación e impresión de licencias y certificados de titulares, 3) brinda la posibilidad de personalizar los documentos de identificación de los titulares y la gestión de las habilitaciones de los mismos, 4) provee la generación de un conjunto de notificaciones y reportes necesarios, agilizando el acceso a la información deseada. A continuación, se muestran las soluciones brindadas en la nueva versión del sistema a las problemáticas identificadas.

- ✓ Actualización a *Symfony* 3.4 para corregir los siguientes problemas de seguridad encontrados
    1. Posible ataque con contraseñas largas emitidas por el usuario.
    2. Vulnerabilidad de redireccionamiento abierto en los controladores de seguridad.
    3. La protección CSRF[1] (*cross site request forgery*) no usa *tokens* diferentes para los protocolos HTTP[2] y HTTPS[3].
    4. Lectores de paquetes para la extensión php Intl[4] que se salen de camino.
    5. Asegurarse de que los datos enviados son archivos cargados.
    6. Problema de fijación de sesión para la autenticación.

1 Falsificación de petición en sitios cruzados.
2 Protocolo de transferencia de hipertextos.
3 Protocolo de transferencia de hipertextos seguros.
4 Extensión para la internacionalización.

7. Denegación de servicio al utilizar *PDOSessionHandle*r.
8. Los *tokens* CSRF no se borrarán durante el cierre de sesión, lo que permite la reparación del *token* CSRF.
9. Vulnerabilidad abierta en el redireccionamiento en los controladores de seguridad.
10. Eliminar soporte para encabezados HTTP heredados y riesgosos.
11. Divulgación de la ruta completa de archivos cargados.
12. Vulnerabilidad abierta en el redireccionamiento al inicio de sesión.

- ✓ Nuevo paginado para que las consultas a la base de datos sean más rápidas y cargue menos datos.
- ✓ Nuevos reportes y filtros que se generan en tablas en la aplicación y se exportan a pdf.
  1. Reporte de titulares por tipo de licencias, activos y pasivos.
  2. Reporte de los pilotos que diga cuántos son capitanes y cuántos son copilotos (en general y por tipo de aeronave).
  3. Reporte de cantidad de titulares por edades según el rango que sea seleccionado por tipo de licencias.
  4. Reporte de quienes están en edad de jubilación por tipo de licencias.
  5. Reporte de cantidad de hombres y mujeres en general y por tipo de licencias.
  6. Reporte de cantidad de titulares por empresas según los tipos de licencias.
  7. Reporte del total de titulares por habilitaciones dentro de los tipos de licencias.

2.2 Modelo conceptual

El modelo conceptual consiste en organizar un esquema teórico de los datos, para su paso de la forma real a la forma correcta para un sistema informático. Esta información es almacenada y gestionada por un sistema gestor de base de datos. En este modelo se encuentran distintos modos de utilizarlos como el modelo entidad-relación y el modelo semántico. (León, 2018)

A continuación, en la figura 1 se muestra el modelo conceptual que sirve como técnica de análisis de requisitos y de diseño de la base de datos.

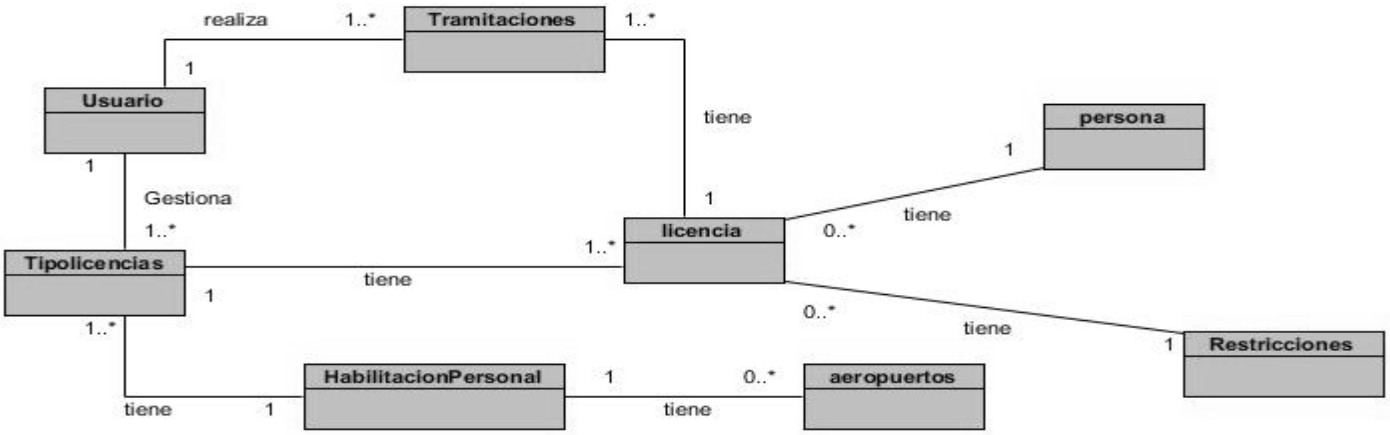


Figura 1. Modelo conceptual

(Fuente: Elaboración propia)

Descripción de los elementos del modelo conceptual

- ✓ **Licencia:** Entidad que contiene todas las licencias del sistema.
- ✓ **Usuario:** Entidad que contiene a todos los usuarios del sistema.
- ✓ **Habilitación personal:** Entidad que contiene todas las habilitaciones que se le pueden realizar a las licencias.
- ✓ **Tramitación:** Entidad donde los usuarios realizan las tramitaciones de las licencias.
- ✓ **Persona:** Entidad que contiene toda la información de todas las personas del sistema.
- ✓ **Tipo de Licencias:** Entidad que contiene los tipos de licencias.
- ✓ **Aeropuertos:** Contiene los aeropuertos y el estado en que estos se encuentran.
- ✓ **Restricciones:** Contiene las restricciones aplicables a las licencias.

2.3 Requisitos

Los requisitos para un sistema son la declaración de los servicios proporcionados por el sistema y sus restricciones operativas. Estos reflejan la necesidad de los clientes de resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. (Sommerville, 2005)

Las principales técnicas utilizadas durante el proceso de desarrollo para recopilar los requisitos de software fueron las siguientes:

- ✓ **Entrevista:** Se consultaron las personas que utilizan la versión actual del Sistema de Gestión de Licencias.
- ✓ **Revisión de investigaciones:** Se estudiaron textos que abordan la seguridad en *Symfony 2.*

2.3.1 Requisitos funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos los requisitos funcionales tienen la posibilidad de declarar lo que el sistema no puede hacer. Dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque tomado al redactar los requisitos. (Sommerville, 2005)

A continuación, se muestra en la tabla 1 los requisitos de software organizados por agrupación y funcionalidad:

Tabla 1*. Requisitos funcionales por agrupación*

(Fuente: Elaboración propia)

Definir nivel estructural	
Gestionar usuario	**RF 1:** Insertar un usuario
	RF 2: Actualizar un usuario
	RF 3: Borrar un usuario
	RF 4: Listar los usuarios
Gestionar titular	**RF 5:** Insertar un titular
	RF 6: Actualizar un titular
	RF 7: Borrar un titular
	RF 8: Listar los titulares
Gestionar titular externo	**RF 9:** Insertar un titular externo
	RF 10: Actualizar un titular externo

	RF 11: Borrar un titular externo
	RF 12: Listar los titulares externos
Gestionar miembro de tripulación	**RF 13:** Insertar un miembro de tripulación
	RF 14: Actualizar un miembro de tripulación
	RF 15: Borrar un miembro de tripulación
	RF 16: Listar los miembros de tripulación
Iniciar licencia	**RF 17:** Iniciar un tipo de licencia titular
	RF 18: Iniciar un tipo de licencia titular externo
	RF 19: Iniciar un tipo de licencia miembro tripulación
Adicionar habilitación a licencia	**RF 20:** Adicionar habilitación a un titular
	RF 21: Adicionar habilitación a un titular externo
Gestionar licencia	**RF 22:** Renovar una licencia
	RF 23: Convalidar una licencia
	RF 24: Prorrogar una licencia
	RF 30: Exportar una licencia
	RF 25: Generar notificaciones
Exportar documentación	**RF 26:** Exportar Solicitud Jurada de Aspirante a Licencia de iniciación.
	RF 27: Exportar Solicitud Jurada de Aspirante a Licencia de renovación.
	RF 28: Exportar Solicitud Jurada de Aspirante a Licencia de añadir habilitación.
	RF 29: Exportar Solicitud Jurada de Aspirante a Licencia de prorrogar.

	RF 31: Exportar Solicitud Jurada de Aspirante a Licencia para Convocatoria.
	RF 32: Exportar un certificado de validez.
	RF 33: Exportar un certificado a miembro de tripulación.
Gestionar centro de instrucción	**RF 34:** Insertar un centro de instrucción.
	RF 35: Actualizar un centro de instrucción.
	RF 36: Borrar un centro de instrucción.
	RF 37: Listar los centros de instrucción.
	RF 38: Generar reporte de los centros de instrucción.
Gestionar programa de instrucción	**RF 39:** Insertar un programa de instrucción.
	RF 40: Actualizar un programa de instrucción.
	RF 41: Borrar un programa de instrucción.
	RF 42: Listar los programas de instrucción.
	RF 43: Generar certificado que avala el programa de instrucción.
Gestionar simulador de vuelo	**RF 44:** Insertar un simulador de vuelo.
	RF 45: Actualizar un simulador de vuelo.
	RF 46: Borrar un simulador de vuelo.
	RF 47: Listar los simuladores de vuelo.
	RF 48: Generar certificado que avala el simulador de vuelo.
Generar reportes	**RF 49:** Reporte de trámites realizados por meses y anual.
	RF 50: Reporte de estadística de la OACI.
	RF 51: Reporte de horas de vuelo de un tripulante técnico en un período de tiempo y por tipos de avión.

	RF 52: Reporte de titulares por tipo de licencias, activos y pasivos.
	RF 53: Reporte de los pilotos que diga cuantos son capitanes y cuantos son copilotos (en general y por tipo de aeronave).
	RF 54: Reporte de cantidad de titulares por edades según el rango que sea seleccionado por tipo de licencias.
	RF 55: Reporte de quienes están en edad de jubilación por tipo de licencias.
	RF 56: Reporte de cantidad de hombres y mujeres en general y por tipo de licencias.
	RF 57: Reporte de cantidad de titulares por empresas según los tipos de licencias.
	RF 58: Reporte del total de titulares por habilitaciones dentro de los tipos de licencias.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente a penas se aplican a características o servicios individuales del sistema. (Sommerville, 2005)

Los requisitos no funcionales son restricciones que deben cumplirse para lograr que el sistema desarrollado logre su función. Estas restricciones son de diferentes tipos. A continuación, se muestran los requisitos no funcionales del sistema:

Seguridad

- ✓ **RNF 1:** El acceso al sistema sólo puede realizarse para un usuario registrado dado.
- ✓ **RNF 2:** La modificación o borrado de datos solo podrá ser realizada por administradores del sistema.

Mantenibilidad

- ✓ **RNF 3:** El sistema deberá estar completamente documentado, tanto en manuales de usuarios, de instalación y de administración como en el código fuente de la aplicación.

Diseño

- ✓ **RNF 4:** El lenguaje de programación a utilizar para el sistema de gestión de licencias tiene que ser PHP 5.5.9 o superior.

Software

- ✓ **RNF 5:** En el entorno donde se va a desplegar el sistema de gestión de licencias, tiene que estar instalado *PostgreSQL*[5] en la versión 9.1 o superior.

Rendimiento

- ✓ **RNF 6:** Los tiempos de respuesta y velocidad de procesamiento de la información serán menores de 5 segundos.

Soporte

- ✓ **RNF 7:** El componente contará antes de su puesta en marcha con un período de pruebas, se le dará mantenimiento, configuración y se brindará el servicio de instalación.

Verificación de datos

- ✓ **RNF 8*:*** El sistema deberá validar la información que se introduzca en cualquiera de los formularios. Esta validación incluye la obligatoriedad de los campos, el tipo y sus características.

2.4 Historias de usuario

Las historias de usuario (HU) son requisitos ya que expresan el problema que el sistema o producto software debe resolver. Son un enfoque de requisitos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad. (Izaurralde, 2013)

Se tiene en cuenta lo expresado por Isaurralde al describir las HU y se agrega que son artefactos que posibilitan entender de una manera más detallada el funcionamiento de cada uno de los requisitos funcionales de un software. Describe su funcionamiento y también condiciones que deben cumplirse para dar solución a un requisito. Posibilitan además al programador que se designe para realizar mantenimiento a la aplicación desarrollada, nutrirse de conocimiento relacionado al funcionamiento interno de cada

5 *Sistema de gestión de bases de datos relacional orientado a objetos; software libre, publicado bajo la licencia PostgreSQL.*

funcionalidad. Se decide utilizar el artefacto HU para describir los requisitos de software, haciendo uso del escenario número cuatro de la metodología AUP-UCI y se describe una HU por cada requisito funcional. El negocio está bien definido y se cuenta con la presencia del cliente durante el proceso de desarrollo. En las tablas 2 y 3 que se muestran a continuación se describen dos HU y en el *anexo 3* se pueden visualizar las restantes.

Tabla 2. *Historia de usuario: Gestionar usuario*

(Fuente: Elaboración propia)

Historia de Usuario	
Nombre: Gestionar usuario	**Número:** HU_1
Modificaciones: Ninguna	
Programador: Yunier Pérez Bejerano	**Iteración Asignada:** 1
Prioridad en el negocio: Media	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos Reales:**
Descripción: El sistema permite insertar, modificar, eliminar listar usuarios en el sistema.	
Observaciones:	
Prototipo de interfaz de usuario:	

Tabla 3. *Historia de usuario: Gestionar personal*

(Fuente: Elaboración propia)

Historia de Usuario	
Nombre: Gestionar personal	**Número:** HU_2
Modificaciones: Ninguna	

Programador: Yunier Pérez Bejerano	**Iteración Asignada:** 1
Prioridad en el negocio: Media	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos Reales:**

Descripción: El sistema permite insertar, modificar y eliminar los datos del personal.
Del personal se registran los siguientes datos:

- ✓ Tipo de personal (Titular, Titular Externo, Miembro de Tripulación)
- ✓ Nombre (Valor alfanumérico)
- ✓ Apellidos (Texto)
- ✓ Nombre del padre (Texto)
- ✓ Nombre de la madre (Texto)
- ✓ Dirección (Valor alfanumérico)
- ✓ Carnet de Identidad (Valor numérico de 11 dígitos)
- ✓ Estado Civil (Casado, No Casado)
- ✓ Teléfono fijo (Valor numérico)
- ✓ Teléfono celular (Valor numérico)
- ✓ Nacionalidad (Valor de solo letras)
- ✓ Ciudadanía (Valor solo letras)
- ✓ País (Valor de solo letras)
- ✓ Pasaporte (Valor alfanumérico)
- ✓ Fecha de nacimiento (Fecha(dd/mm/aa))
- ✓ Estatura (Valor numérico (cm))
- ✓ Peso (Valor numérico (kg))
- ✓ Sexo (Valor de solo letras (F, M))
- ✓ Estado (Activo, Pasivo)
- ✓ Color de Ojos (NEGROS, PARDOS, VERDES, AZULES)
- ✓ Pelo (NEGRO, CASTAÑO, RUBIO, ROJO, BLANCO, CANOSO, CALVO)
- ✓ Nivel Escolar (PRIMARIA, SECUNDARIA, PREUNIVERSITARIO, TÉCNICO MEDIO, UNIVERSITARIO)
- ✓ Ocupación (INSPECTOR, A. S. P. A VUELOS, AEROMOZA, SOBRECARGO,

DESPACHADOR, DIRECTIVO, ESPECIALISTA, INSTRUCTOR, MEC. A BORDO, NAVEGANTE, PILOTO, TEC. SEG. OP., TECNICO, OTROS)

- ✓ Foto (Valor seleccionable)
- ✓ Empresa a la que pertenece (AEROALBA, AEROCARIBBEAN, AEROGAVIOTA, ALBA, CACSA, CUBANA, CUBANA - BLUE PANORAMA, E.M.I. Y. GAGARIN, ECASA, ENSA, G.A.M. *Technics*, IACC, IBECA)

Observaciones: El sistema permite buscar una foto del titular.

El sistema muestra los siguientes campos en dependencia del tipo de personal:

- ✓ **Titular:**

  Nombre (Valor alfanumérico)

  Apellidos (Texto)

  Nombre del padre (Texto)

  Nombre de la madre (Texto)

  Dirección (Valor alfanumérico)

  Carnet de Identidad (Valor numérico de 11 dígitos)

  Estado Civil (Casado, No Casado)

  Teléfono fijo (Valor numérico)

  Teléfono celular (Valor numérico)

  Nacionalidad (Valor de solo letras)

  Ciudadanía (Valor solo letras)

  Fecha de nacimiento (Fecha(dd/mm/aa))

  Estatura (Valor numérico (cm))

  Peso (Valor numérico (kg))

  Sexo (Valor de solo letras (F, M))

  Estado (Activo, Pasivo)

  Color de Ojos (NEGROS, PARDOS, VERDES, AZULES)

  Pelo (NEGRO, CASTAÑO, RUBIO, ROJO, BLANCO, CANOSO, CALVO)

  Nivel Escolar (PRIMARIA, SECUNDARIA, PREUNIVERSITARIO, TÉCNICO MEDIO, UNIVERSITARIO)

Ocupación (INSPECTOR, A. S. P. A VUELOS, AEROMOZA, SOBRECARGO, DESPACHADOR, DIRECTIVO, ESPECIALISTA, INSTRUCTOR, MEC. A BORDO, NAVEGANTE, PILOTO, TECNICO, OTROS)

Foto (Valor seleccionable)

Empresa a la que pertenece (AEROALBA, AEROCARIBBEAN, AEROGAVIOTA, ALBA)

- ✓ **Titular Externo:**

  Nombre

  Nacionalidad

  Pasaporte

  Fecha de nacimiento (Valor seleccionable)

  Sexo (Valor seleccionable)

  Foto (Valor seleccionable)

  Ocupación (AEROMOZA, SOBRECARGO, DIRECTIVO, ESPECIALISTA, INSTRUCTOR, PILOTO, TECNICO, OTROS)

- ✓ **Miembro de Tripulación:**

  Nombre

  Dirección

  Nacionalidad

  Sexo (Valor seleccionable)

  Fecha de Nacimiento (Valor seleccionable)

  Ocupación (INSPECTOR, A. S. P. A VUELOS, AEROMOZA, SOBRECARGO, DESPACHADOR, DIRECTIVO, ESPECIALISTA, INSTRUCTOR, MEC. A BORDO, NAVEGANTE, PILOTO, TECNICO, OTROS)

  Foto

  El campo estado pasivo son casos tales como por ejemplo muerte, deserción o cualquier otra causa similar.

Prototipo de interfaz de usuario:

2.5 Arquitectura del sistema

El patrón arquitectónico seleccionado en la presente investigación para el desarrollo de la propuesta de solución es Modelo-Vista-Controlador (*MVC*), ya que es el que utiliza el *framework* de desarrollo ***Symfony 3.4***. Entre sus ventajas está la posibilidad de separar la lógica de negocio (el modelo) y la presentación (la vista), por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como en un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones. El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

Symfony 3.4 implementa el patrón arquitectónico MVC como se muestra en la siguiente figura:

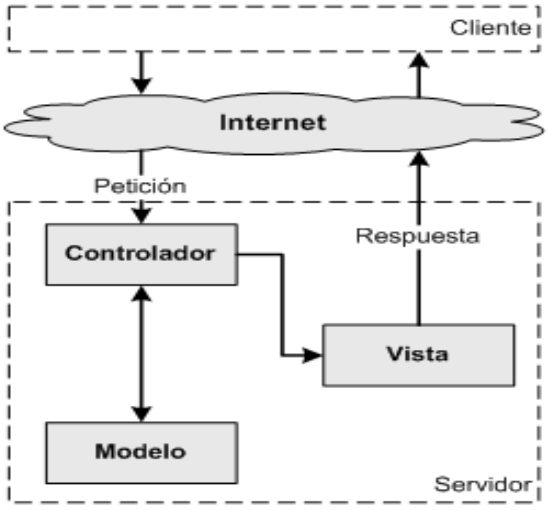


Figura 2. Patrón arquitectónico
(Fuente: *(Uniwebsidad, 2018)*)

Symfony 3.4 simplifica el proceso de acceder a cada parte de la aplicación, obtiene lo mejor del patrón arquitectónico MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. En primer lugar, el controlador frontal y el *layout* son comunes para todas las acciones de la aplicación. Se

pueden tener varios controladores y varios *layouts*, pero solamente es obligatorio tener uno de cada uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que ***Symfony 3.4*** lo genera de forma automática. (Uniwebsidad, 2018)

2.6 Diagrama de clases de la solución

Un diagrama de clases es una herramienta para comunicar el diseño de un programa que se creó para orientar objetos y que permite modelar relaciones entre diferentes entidades. Los diagramas de clase describen la vista estática del modelo o parte del modelo, describiendo que atributos y comportamientos tienen en lugar de detallar los métodos para realizar operaciones, son más útiles para ilustrar relaciones entre clases e interfaces. Las generalizaciones, agregaciones, y asociaciones son todas valiosas al reflejar herencias, composición o uso, y conexiones respectivamente (Gómez & Olive, 2003). En *anexo 2* se muestra el diagrama de clases general para la solución propuesta.

Para las aplicaciones web se realizan los diagramas de clases de diseño con estereotipos web para cada historia de usuario. A continuación, se muestra en la tabla 4 el conjunto de estereotipos que se pueden asociar a las clases y establecer las relaciones entre estas para representar una aplicación web.

Tabla 4*. Estereotipos web para UML*

(Fuente: Elaboración propia)

Estereotipos para las clases	
Estereotipo	**Descripción**
 *Client Page*	Representan páginas que son dibujadas por el navegador web y pueden ser una combinación de algún o algunos lenguajes de marcado, *scripts* del lado del cliente, islas de datos, etc.
 Server Page	Representa una página web que tiene *scripts* ejecutados por el servidor. Estos *scripts* interactúan con los recursos que se encuentran al alcance del servidor. Sólo puede mantener relaciones con objetos que se encuentren en el servidor.

 Form	Representa una colección de campos de entrada que forman parte con una página del lado cliente (*Client Page*). Tiene una correspondencia directa con la etiqueta <*FORM*> de *HTML*.
Estereotipos para las Relaciones entre las Clases	
Link	Representa un apuntador desde una "*client page*" hacia una "*client page*" o "*server page*". Corresponde directamente con una etiqueta <a> de *HTML.*
Submit	Esta relación siempre se da entre una "*form*" y una "*server page*", por supuesto, la "*server page*" procesa los datos que la "*form*" le envía (*submits*).
Build	Sirve para identificar cuales "*server page*" son responsables de la creación de una "*client page*". Una "*server page*" puede crear varias "*client page*", pero una "*client page*" sólo puede ser creada por una sola "*server page*". Esta relación siempre es unidireccional.
Redirect	Esta es también una relación unidireccional que indica que una página web redirige hacia otra. En caso de que la página origen sea una "*client page*" esta asociación corresponderá con la etiqueta "*META*" y valor HTTP-EQUIV de "*Refresh*".

A continuación, en la figura 3 se muestra un diagrama de clases de diseño con estereotipos web para la historia de usuario: Adicionar Titular.

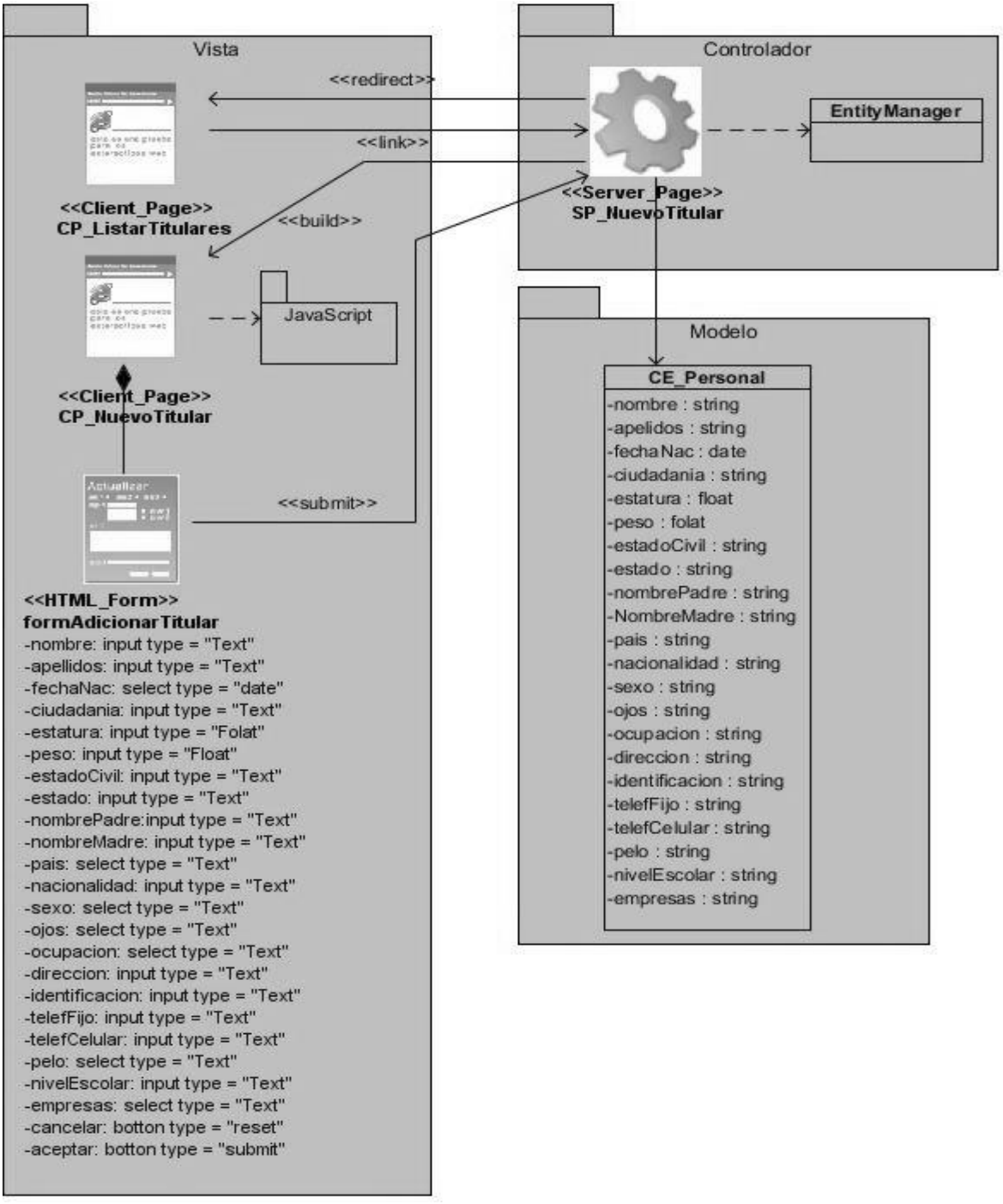


Figura 3. Diagrama de clases del diseño de la HU Adicionar Titular

(Fuente: Elaboración propia)

2.7 Patrones de diseño

Los patrones de diseño son técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. También son propuestas generales planteadas en término de una colaboración de objetos mediante las que se resuelven una gran cantidad de problemas que aparecen una y otra vez en el desarrollo de aplicaciones informáticas. Se consideran una serie de buenas prácticas de aplicación recomendable en el diseño de software. (Iturralde, 2016)

2.7.1 Patrones GRASP

En diseño orientado a objetos, **GRASP** son patrones generales de software para asignación de responsabilidades, es el acrónimo de "GRASP (*General Responsibility Assignment* Software *Patterns*)". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software. La asignación eficiente de responsabilidades a los componentes del software es la actividad más importante en el análisis y diseño orientado a objetos (Iturralde, 2016). Entre los patrones existentes se seleccionaron los que se describen a continuación, ya que fundamentan cuestiones y aspectos importantes del diseño en la propuesta de solución.

Experto: El GRASP de experto en información es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento) *(Iturralde, 2016)*. A continuación, en la siguiente figura 4 se muestra la utilización del patrón experto en la solución, esta figura es un fragmento del diagrama de clases de la solución. Se evidencia el uso de este patrón en todas las clases de la propuesta de solución ya que cada una conoce su información e implementa sus funcionalidades.

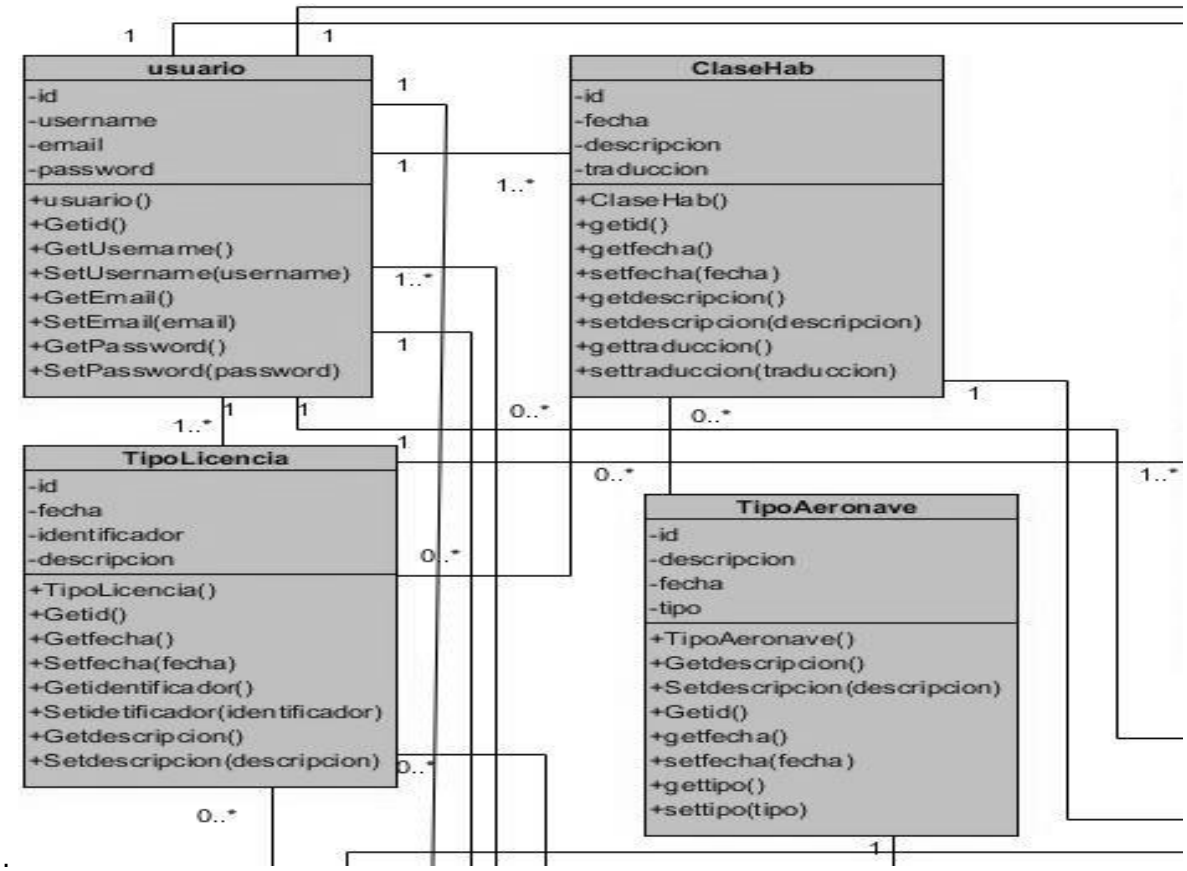


Figura 4. Ilustración del patrón experto

(Fuente: Elaboración propia)

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. (Iturralde, 2016)

La nueva instancia deberá ser creada por la clase que:

- ✓ Tiene la información necesaria para realizar la creación del objeto.
- ✓ Usa directamente las instancias creadas del objeto.
- ✓ Almacena o maneja varias instancias de la clase.
- ✓ Contiene o agrega la clase.

A continuación, se muestra el uso del patrón creador en la clase *AeropuertoController* en el método *createAction*, al crear instancias de la clase entidad Aeropuerto, como se muestra en la Figura 5 en la tercera línea de código.

```
public function createAction(Request $request, HabilitacionPersonal $habpers, $proceso) {
    $session = $request->getSession();
    $entity = new Aeropuerto();
    $form = $this->createForm(new AeropuertoType(), $entity);
```

Figura 5. Ilustración del patrón creador

(Fuente: Elaboración propia)

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. (Iturralde, 2016)

En la figura 6 se muestra el uso del patrón controlador en la solución propuesta.

```
class AeropuertoController extends Controller
{
    public function indexAction($id_habpers, $proceso)
    {
        $em = $this->getDoctrine()->getManager();
        $habilitacionPersonal = $em->getRepository('AppBundle:Habilitacion\HabilitacionPersonal')->find($id_habpers);
        return $this->render('AppBundle:Aeropuerto:list.html.twig', array(
            'id_habpers' => $id_habpers,
            'proceso' => $proceso,
            'habilitacion' => $habilitacionPersonal
        ));
    }
```

Figura 6. Ilustración del patrón controlador

(Fuente: Elaboración propia)

Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (Iturralde, 2016)

1. **Acoplamiento de Contenido**: Cuando un módulo referencia directamente el contenido de otro módulo. (En lenguajes de alto nivel es muy raro).
2. **Acoplamiento Común**: Cuando dos módulos acceden (y afectan) a un mismo valor global.
3. **Acoplamiento de Control**: Cuando un módulo le envía a otro un elemento de control que determina la lógica de ejecución del mismo.

En la Figura 7 se muestra el uso de patrón bajo acoplamiento en la solución.

```
class AeropuertoController extends Controller
{
    public function indexAction($id_habpers, $proceso)
    {
        $em = $this->getDoctrine()->getManager();
        $habilitacionPersonal = $em->getRepository('AppBundle:Habilitacion\HabilitacionPersonal')->find($id_habpers);
```

Figura 7. Ilustración del patrón bajo acoplamiento
(Fuente: Elaboración propia)

2.7.2 Patrones GOF (Gang of Four)

Los patrones GOF describen soluciones simples y elegantes a problemas específicos en el diseño de software. Se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. También representan soluciones técnicas basadas en la programación orientada a objetos que favorecen la reutilización del código. (Iturralde, 2016)

Decorador: Añade funcionalidad a una clase dinámicamente. Esto permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera. Se aplica con la intención de proporcionar una forma flexible de introducir o eliminar funcionalidad de un componente sin modificar su apariencia externa o su función. (Iturralde, 2016)

Se evidencia el uso en la clase *HabilitacionPersonalController* en la funcionalidad *editAction()* a la hora de modificar los atributos de un objeto clase existente, sin cambiar la apariencia como se muestra en la figura 8.

```
$entity->setClase($em->getRepository('AppBundle:Habilitacion\Clase')->find($clase));
$entity->setTipo($em->getRepository('AppBundle:Habilitacion\Tipo')->find($tipo));
$entity->setFecha(new \DateTime($form->get('fecha')->getData()));
$em->persist($entity);
$em->flush();
```

Figura 8. Ilustración del patrón decorador

(Fuente: Elaboración propia)

Fachada: Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. Es un tipo de patrón de diseño estructural. Viene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre estos. Este patrón se utiliza en la definición de las plantillas usando el motor de plantillas *TWIG*. Permite decorar la presentación de la información al usuario. (Iturralde, 2016)

A continuación, en la Figura 9 se muestra el uso del patrón fachada de la solución propuesta.

```
<strong>Nuevo Centro de Instrucción</strong>
<form action="{{ path('centro_instruccion_create') }}" {{ form_enctype(form) }} method="post"
    {{ form_start(form) }}
    <br/>
    <div class="row-fluid">
        <div class="span12">
            <div class="span6">
                <div class="control-group">
                    <label for="inputUsername" class="control-label">Número</label>

                    <div class="controls" style="...">
                        {{ form_widget(form.numero) }}
                        {{ form_errors(form.numero) }}
                    </div>
                </div>
                <div class="control-group">
                    <label for="inputUsername" class="control-label">Nombre</label>

                    <div class="controls" style="...">
                        {{ form_widget(form.entidad) }}
                        {{ form_errors(form.entidad) }}
                    </div>
                </div>
                <div class="control-group">
                    <label for="inputEmail" class="control-label">Dirección</label>
```

Figura 9. Ilustración del patrón fachada.

(Fuente: Elaboración propia)

2.8 Modelo de datos

Un modelo de datos es una parte esencial en el proceso de modelado de una base de datos. Este es una estructura abstracta que documenta y organiza la información de los datos y las relaciones entre ellos. El propósito de un modelo de datos es, por una parte, representar los datos y por otra, ser comprensible. En él se describen los datos, las relaciones de datos y la semántica de los datos. (Gómez & Olive, 2003)

En el (*Anexo 1 Modelo de datos)* referente al modelo de datos está ilustrado el diagrama entidad relación de la solución propuesta.

2.9 Diagrama de despliegue

Un diagrama de despliegue muestra la arquitectura del sistema en el despliegue del software. También es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar la disposición física de los artefactos software en nodos. Muestra la arquitectura del sistema como el despliegue de los artefactos de software a los objetivos de despliegue (Jiménez, 2016). En figura 10 se muestra el diagrama de despliegue de la solución propuesta.

Figura 10. Diagrama de Despliegue

(Fuente: Elaboración propia)

Cliente: Se refiere a las estaciones de trabajo que realizan las peticiones al servidor de aplicaciones donde está hospedado el Sistema de Gestión de Licencias Aeronáuticas del personal aeronáutico del IACC versión 2.0, mediante un navegador web utilizando el protocolo de comunicación *HTTPS* por el puerto 443.

Servidor Web: Es el encargado de brindar la interfaz de la plataforma para que los usuarios puedan hacer uso de esta, almacena todo el código fuente del sistema y se comunica por medio de los protocolos *TCP* con el servidor de bases de datos.

Servidor de Bases de Datos: Almacena toda la información que brinda la plataforma hospedada en el servidor de aplicaciones. La información es obtenida o modificada en dependencia del nivel de privilegio del usuario que realiza la petición. La comunicación con el servidor de aplicaciones es a través del protocolo *TCP* empleando el puerto 5432.

Conclusiones parciales del capítulo

El análisis y diseño de la propuesta permitió realizar una descripción detallada de las características del Sistema de Gestión de Licencias del Personal Aeronáutico 2.0, lo que permitió un mejor entendimiento para la fase de implementación al tener los principales artefactos para el desarrollo. Se definieron 58 requisitos funcionales y 8 no funcionales los cuales proporcionan una guía de desarrollo de las funcionalidades del sistema.

Se seleccionó la arquitectura MVC para el desarrollo de la solución ya que esta responde al desarrollo del sistema.

La definición del estilo arquitectónico y la realización de los diagramas de clases permitieron un mejor entendimiento del funcionamiento del sistema.

Se seleccionaron los patrones *GRASP* y *GoF* a utilizar en la implementación para lograr una adecuada reutilización del código para futuras versiones.

Capítulo 3. Implementación y prueba de la solución

Introducción

En el presente capítulo se lleva a cabo la implementación y prueba de la solución, esta etapa es de gran importancia en el proceso de desarrollo de software ya que se desarrolla cada componente del sistema y se realizan las pruebas para comprobar que se implementó la solución propuesta con los requisitos necesarios.

3.1 Implementación y prueba de la solución

La fase de implementación y prueba es el período durante el cual se implementa la solución y se comprueba que es estable y utilizable. Como resultado en esta fase se obtendrá una aplicación que cumpla con las funciones necesarias para dar solución al problema existente en el Sistema de Gestión de Licencias del Personal Aeronáutico del Instituto de la Aeronáutica Civil de Cuba.

3.2 Estructura interna de la solución

La estructura de directorios de *Symfony* varía según el tipo de proyecto que hallamos creado (*framework-standard-edition, website-skeleton*). Para el Sistema de Gestión de Licencias del Instituto de la Aeronáutica Civil de Cuba versión 2.0, la estructura interna esta complementada por el *framework* de desarrollo web *Symfony* en su versión 3.4 de la siguiente manera:

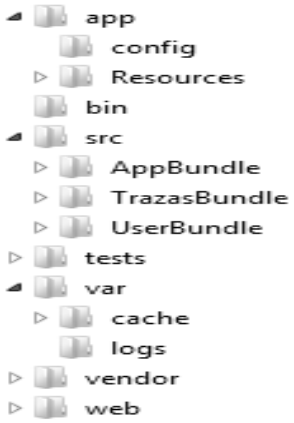


Figura 11. Directorio de Symfony 3.4
(Fuente: Estructura interna de Symfony)

La figura 11 muestra:

- ✓ ***app/config*:** Contiene todos los ficheros de configuración del proyecto, por ejemplo, los archivos rounting.yml y security.yml.
- ✓ **app/Resources:** En esta ruta se tienen las plantillas *TWIG* y los ficheros de traducción necesarios para la presentación de la aplicación.
- ✓ **bin:** Los binarios del proyecto se ubican aquí, por ejemplo, la consola que se utiliza, posteriormente, para las llamadas a doctrine.
- ✓ **src/AppBundle:** Este directorio contiene el código relativo a los controladores, rutas, vistas y formularios. Es decir, el código específico de la lógica de negocio de la aplicación.
- ✓ **src/TrazasBundle:** Este directorio contiene todo el código necesario para realizar las trazas del sistema.
- ✓ **src/UserBundle:** Este directorio contiene el código de la gestión y administración de usuarios del sistema.
- ✓ **tests:** Contiene las aplicaciones de test al sistema.
- ✓ **var/cache:** Almacena todos los archivos de caché generados por la aplicación.
- ✓ **var/logs:** Almacena todos los archivos de *logs* generados por la aplicación.
- ✓ **vendor:** En este directorio se encuentran las dependencias de código de la aplicación.
- ✓ **web:** En el directorio se almacenan los controladores frontales y todas las hojas de estilo, ccs, *JavaScript*, imágenes entre otras. *(SensioLabs, 2019)*

3.3 Estándares de codificación

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible (Arias, 2019). Partiendo de lo antes expuesto se definen los siguientes estándares para la solución propuesta:

Nomenclatura de las clases: Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se emplea notación *PascalCasing*, esta especifica que los identificadores y nombres de variables, métodos y funciones que están compuestos por múltiples palabras juntas, deben iniciar cada palabra con letra mayúscula, lo que posibilita que con sólo leerlo se reconozca el propósito de la misma.

Ejemplo: "*HabilitacionPersonal"* en este caso el nombre de clase está compuesto por dos palabras iniciadas cada una con letra mayúscula.

Clases controladoras: Los nombres de las clases controladoras comienzan con la primera letra en mayúscula seguido por la palabra "*Controller*". En caso de que sea un nombre compuesto comienza con mayúsculas los dos nombres acompañado de la palabra "*Controller*". **Ejemplo:** "*CentroInstruccionController*".

Clases repositorios: Las clases que se encuentran dentro de *Repository* comienzan con mayúscula y después del nombre llevan la palabra "*Repository*". **Ejemplo:** "*AvionRepository*".

Clases entidades: Su nombre comienza con mayúsculas y están ubicadas dentro del directorio *Entity* del proyecto. Son las clases que crean el esquema de la base de datos según su contenido. **Ejemplo:** "Licencia".

Clases formularios: Su nombre comienza con mayúsculas y están ubicadas dentro del directorio *Form* del proyecto. **Ejemplo:** "*ClaseType*".

Nomenclatura de las funcionalidades o métodos

El nombre de los métodos está descrito con la inicial del identificador en minúscula. **Ejemplo:** "*index".*

Además de cumplir con el estándar anterior, los nombres de los métodos están seguidos de la palabra *"Action"* **Ejemplo:** *"indexAction ()".*

A continuación, se explica cómo está estandarizada la estructura interna de las funcionalidades o métodos.

Importaciones: Las importaciones están en líneas separadas.

Codificaciones: Utilizar la codificación UTF-8.

Comentarios: Los comentarios deben ser lo suficientemente claros y concisos para que se entienda el propósito de lo que se está desarrollando. En caso de ser una función complicada se debe comentar su interior para lograr una mejor comprensión del código.

Operadores: Los operadores deben tener un espacio a la izquierda y un espacio a la derecha.

Estilo de indentado o sangrado:

- ✓ La declaración de la clase debe estar en el borde izquierdo del área de trabajo.
- ✓ Debe haber una línea en blanco por debajo de la declaración de clases, también de las funcionalidades o métodos.
- ✓ La declaración del nombre de las funcionalidades o métodos debe comenzar después de 4 espacios del borde izquierdo.
- ✓ Las instrucciones que pertenecen a un mismo bloque, dígase una estructura *if-else, for, foreach, case,* iniciarán en la siguiente línea después de la declaración de la estructura y deben comenzar después de cuatro espacios partiendo de la declaración del bloque.
- ✓ Las llaves que abren un bloque de instrucciones deben ir en la misma línea en que se declara el mismo y después de un espacio en blanco.
- ✓ Las llaves que cierran un bloque de instrucciones deben ir en la línea siguiente a la culminación del mismo, y deben cumplir el mismo indentado de la declaración de dicho bloque.

Variables:

- ✓ El nombre de las variables o constantes debe comenzar con minúsculas.
- ✓ Si el nombre de las variables está compuesto por dos palabras o más, se empleará el tipo ***lowerCamelCase*** de la notación ***CamelCasing.***
- ✓ En las funciones que para su ejecución necesiten valores como parámetros, en caso de que sea más de un parámetro, estos deben ir separados por coma (,) y con un espacio después de la coma.

3.4 Diagrama de componentes

El diagrama de componentes muestra las relaciones estructurales entre los componentes de un sistema. Los componentes se consideran unidades autónomas encapsuladas dentro de un sistema o subsistema que proporcionan una o más interfaces. Estos diagramas son generalmente dirigidos al personal de aplicación de un sistema, además presenta una comprensión temprana del sistema global que se está construyendo. (IBM, 2004)

En la Figura 12 a continuación se muestra el diagrama de componentes de la solución.

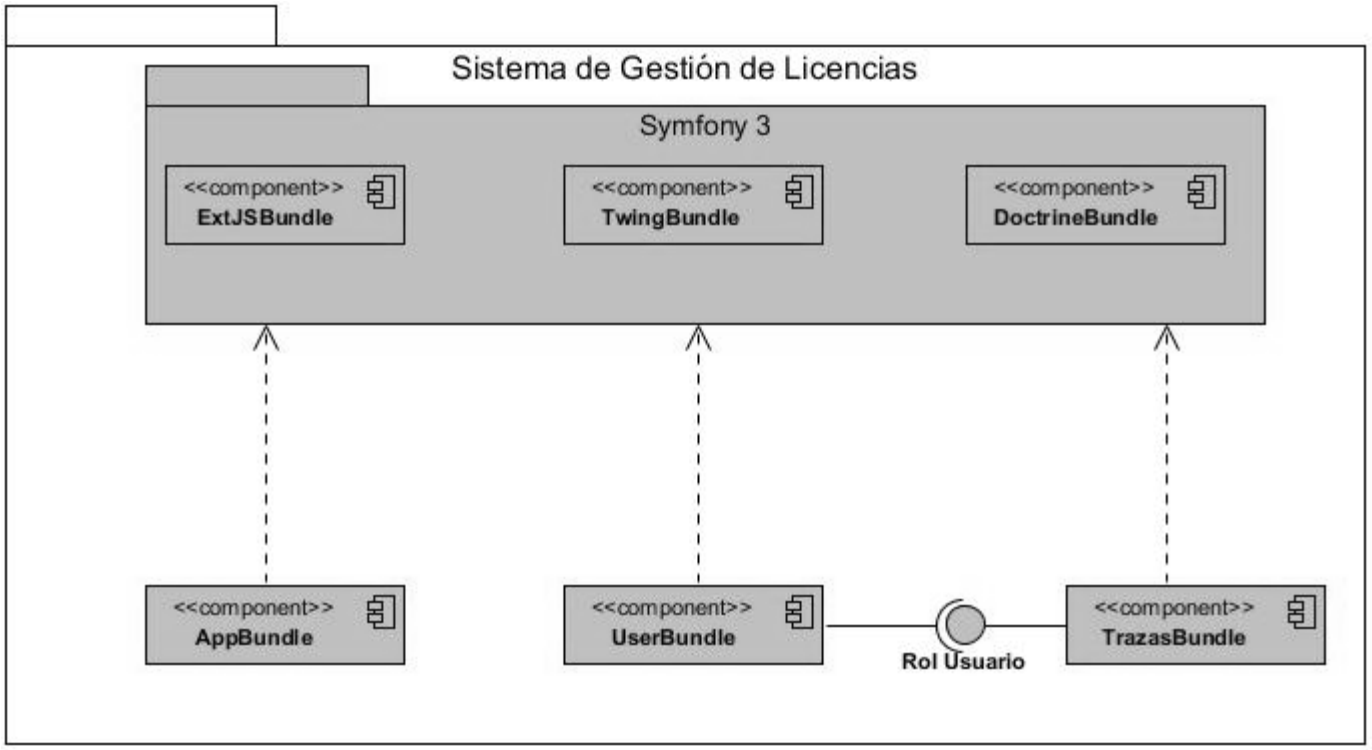


Figura 12. Diagrama de componentes

(Fuente: Elaboración propia)

3.5 Estrategias de pruebas de software

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuando se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requieren. Por tanto, cualquier estrategia de 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. Una estrategia de prueba de software debe ser lo suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto. (Pressman, 2010)

3.5.1 Pruebas de rendimiento

La prueba de rendimiento se diseña para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado de rendimiento de software. Se realizan para medir la respuesta

de la aplicación a distintos volúmenes de carga esperados, se centra en determinar la velocidad con la que el sistema bajo pruebas realiza una tarea en las condiciones particulares del escenario de pruebas. (Sommerville, 2005)

Carga: Este es el tipo más sencillo de pruebas de rendimiento. Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Si la base de datos, el servidor de aplicaciones, etc. También se monitorizan, entonces esta prueba puede mostrar el cuello de botella en la aplicación. (Sommerville, 2005)

Estrés: Esta prueba se utiliza normalmente para romper la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada. (Sommerville, 2005)

Las pruebas de carga y estrés se realizaron con la ayuda de la herramienta *Apache JMeter v5.1.1*[6]*,* la cual permite medir el rendimiento de los recursos como servicios web, lenguajes dinámicos y bases de datos, entre otros.

Estas se llevaron a cabo en un ambiente seleccionado utilizando un ordenador con las siguientes características:

Hardware de prueba (PC cliente):

- ✓ Tipo de procesador: Intel(R) Core(TM) i3-4030U CPU @ 1.90GHz.
- ✓ RAM: 4 GB DDR3.
- ✓ Tipo de Red: Ethernet 10/100Mbps.

Hardware de prueba (PC servidor):

- ✓ Tipo de procesador: Intel(R) Core (TM) i3-4030U CPU @ 1.90GHz
- ✓ RAM: 4 GB DDR3.

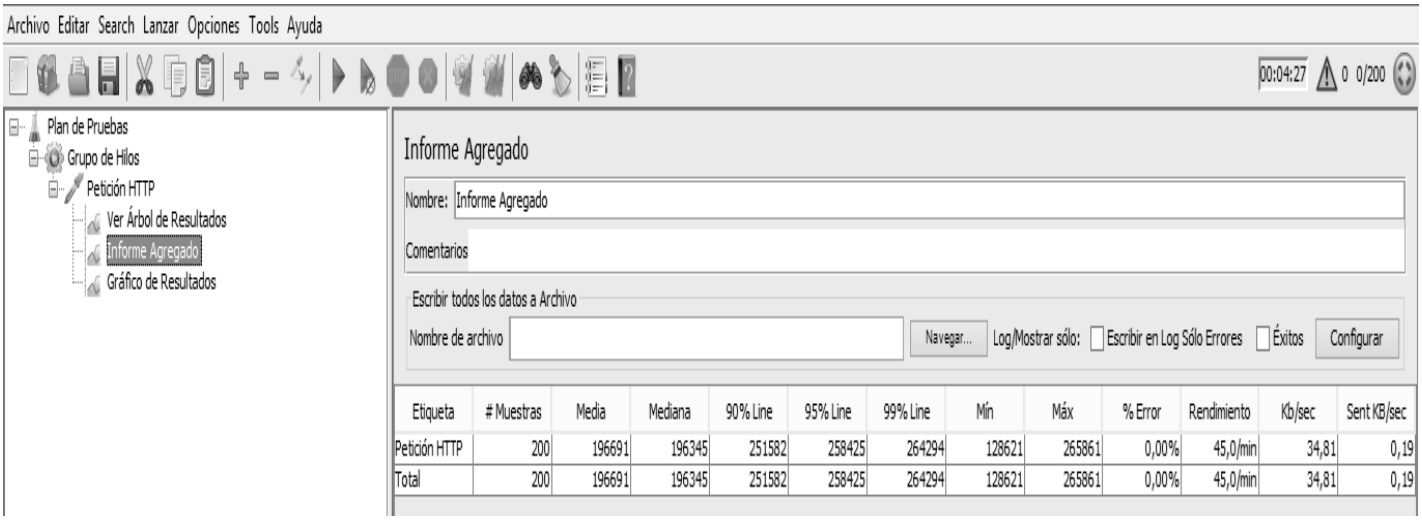
6 http://jmeter.apache.org/

- ✓ Tipo de Red: Ethernet 10/100Mbps.

Software *instalado* en ambas PC:

- ✓ Plataforma: SO Windows 8.1 Pro (PC servidor) y SO Windows 8.1 Pro (PC cliente).
- ✓ Tipo de Sistema: Sistema operativo de 64 bits, procesador x64
- ✓ Servidor de BD: PostgresSQL9.4

A continuación, se muestra en la Figura 13 los resultados obtenidos por la herramienta *Apache JMeter v5.1.1* con un total de doscientos usuarios conectados concurrentemente.



Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
Petición HTTP	200	196691	196345	251582	258425	264294	128621	265861	0,00%	45,0/min	34,81	0,19
Total	200	196691	196345	251582	258425	264294	128621	265861	0,00%	45,0/min	34,81	0,19

Figura 13. Resultado de las pruebas de carga y estrés
(Fuente: Elaboración propia)

Las pruebas de rendimiento arrojaron como resultado que el tiempo total de las solicitudes es de 196345 milisegundos, realizándose con un total de 200 muestras. El tiempo total para los 200 hilos se calcula de la siguiente manera:

Tiempo Total= #Muestras * Media = 200 * 196345 = 39269000 milisegundos

El tiempo promedio requerido por cada hilo se calcula de la siguiente manera:

Tiempo Promedio = (Media / 1000) / Cantidad de Hilos = (196345 / 1000) /200 = 0.981725 segundos

3.5.2 Pruebas de integración

Las pruebas de integración son una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera individual y construir una estructura de programa que se haya dictado por diseño. (Pressman, 2010)

Las pruebas de integración se le realizaron al sistema de gestión de licencias para verificar la compatibilidad y el funcionamiento de las interfaces y módulos lo que arrojó como resultado que cada uno de los componentes y módulos del sistema de gestión de licencias se integran correctamente. También se aplicó para verificar que los elementos de software que interactúan entre si funcionan de manera correcta, en el caso de la versión 2.0 del sistema de gestión de licencias se verifica que los nuevos reportes requeridos por el cliente se integran correctamente al sistema, además de cada una de sus funcionalidades por separado. Las pruebas de integración arrojaron como resultado que existe una correcta integración entre los componentes internos del sistema.

3.5.3 Pruebas de aceptación

Se aplica este tipo de prueba con el objetivo de determinar, por parte del cliente, la aceptación o rechazo del sistema desarrollado. Este tipo de prueba es realizada por el usuario final en lugar de los ingenieros de software. Una prueba de aceptación puede variar desde una prueba de conducción informal hasta una serie de pruebas planificadas y ejecutadas sistemáticamente. La prueba de aceptación puede realizarse durante un periodo de semanas o meses, y mediante ella descubrir errores acumulados que con el tiempo puedan degradar el sistema. (Pressman, 2010)

Entre los tipos de pruebas de aceptación que existen en este caso en particular se realizan las pruebas **alfa** que son las que se desarrollan en la propia compañía desarrolladora del software con el equipo de pruebas del software. Las pruebas de aceptación se les realizan a todas las funcionalidades del sistema. A continuación, se muestra un ejemplo de un caso de prueba realizado, en la tabla 4, los campos de la tabla son los siguientes:

- ✓ **Código:** identificador de la prueba realizada sugerente a la HU a la que hace referencia.
- ✓ **Nombre:** nombre de la prueba a realizar.

- ✓ **Nombre del probador:** nombre de la persona que realiza la prueba.
- ✓ **Descripción:** se describe la funcionalidad que se desea probar.
- ✓ **Condiciones de Ejecución:** muestra las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.
- ✓ **Entradas/Pasos de Ejecución:** descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- ✓ **Resultado esperado:** breve descripción del resultado que se espera obtener con la prueba realizada.
- ✓ **Evaluación de la prueba:** acorde al resultado de la prueba realizada se emite una evaluación sobre la misma. Esta evaluación tiene uno de los dos valores que a continuación se describen:
    1. Satisfactorio
    2. Insatisfactorio

Tabla 5. *Caso de prueba de aceptación método: Adicionar Titular*

(Fuente: Elaboración propia)

Caso de prueba aceptación		
Código de caso de prueba: HU_2	**Nombre de historia de usuario:** Adicionar titular.	
Nombre de la persona que realiza la prueba: Yunier Pérez Bejerano		
Descripción de la prueba: Prueba la funcionalidad adicionar titular		
Entrada/Pasos de ejecución: En el menú del sistema se entra en el módulo “Personal Aeronáutico” luego en el submenú a titulares y en la parte de arriba a la derecha hay un botón que dice “Crear nuevo” y ahí empezaría el proceso de adicionar un nuevo titular en el sistema.		
Escenarios:	**Resultados Esperado**	**Evaluación de la prueba:**
EC 1.1 Adicionar titular con los datos correctamente.	Se adiciona el titular, luego se muestra en cartel que dice que se ha	Satisfactoria.

	adicionado correctamente y muestra el listado de los titulares del sistema incluyendo el nuevo insertado.	
EC 1.2 Adicionar un titular introduciendo caracteres inválidos.	Muestra un mensaje según el error encontrado: 1. Si es un campo de número: “Debe proveer un formato correcto”. 2. Si es un campo de letra: “Debe proveer un formato correcto”.	Satisfactoria.
EC 1.3 Adicionar un titular dejando campos vacíos.	Muestra un mensaje según el error encontrado: 1. Si es seleccionable: “Campo obligatorio. Debe seleccionar un elemento”. 2. Si es un campo de texto o de número “Campo obligatorio”.	Satisfactoria.
EC 1.4 Cancelar	Muestra un cartel preguntando ¿Desea salir de este formulario sin guardar los cambios? En caso de dar aceptar regresa a la lista de titulares y en caso de cancelar se mantiene en el formulario.	Satisfactoria.

Las Pruebas de aceptación arrojaron como resultado que no existía ninguna no conformidad por parte del cliente. Los resultados en todos los casos de prueba fueron satisfactorios por lo que se acepta el producto.

3.5.4 Pruebas funcionales

La prueba funcional se centra en comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente, con el objetivo de validar que las funcionalidades implementadas funcionen correctamente y cumplan con los requisitos definidos con anterioridad. Estas pruebas se enfocan solamente en las entradas y salidas del sistema, sin preocuparse de tener conocimiento

de la estructura interna del programa de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, se basan únicamente en los requisitos de software y especificaciones funcionales. Al definir una prueba de caja negra lo principal es identificar los datos de prueba (entradas) y el resultado esperado del sistema al ingresar esos datos, bien sean los datos de salida o algún comportamiento específico. (Pressman, 2010)

Este tipo de pruebas permite encontrar:

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

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

- ✓ **Partición de equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ **Grafos de causa-efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.
- ✓ **Análisis de valor de frontera:** El análisis de valor de frontera conduce a una selección de casos de prueba que revisan los valores de frontera.

En la tabla 5 se muestran las variables para el caso de prueba adicionar tipo de licencia:

Tabla 6. *Variables para el caso de prueba adicionar tipo de licencia*

(Fuente: Elaboración propia)

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Identificador	Campo de texto	No	Se inserta una cadena que identifique el tipo de licencia.
2	Descripción	Campo de texto	No	Se inserta una descripción que identifique el tipo de licencia.
3	Habilitaciones	Campo seleccionable	No	Se selecciona la o las habilitaciones del tipo de licencia.

Se aplicó el método de caja negra a la solución específicamente la técnica de partición de equivalencia con el siguiente caso de prueba y obteniendo como resultado la tabla 6:

Tabla 7. *Caso de prueba adicionar tipo de licencia*

(Fuente: Elaboración propia)

Nombre del requisito	Descripción	Escenarios de pruebas	Flujo del escenario
Adicionar tipo de licencia	El usuario con los permisos correspondientes adiciona un tipo de licencia proporcionando la información necesaria.	EP 1.1 Adicionar un tipo de licencia insertando un identificador, una descripción y una habilitación de forma correcta.	✓ El usuario hace la petición al sistema para adicionar un nuevo tipo de licencia. ✓ En la clase controladora se activa la función para adicionar un nuevo tipo de licencia. ✓ Se muestra una ventana con el formulario para adicionar un nuevo tipo de licencia. ✓ El usuario entra los datos requeridos para adicionar un nuevo tipo de licencia de manera correcta (identificador, descripción y habilitaciones). ✓ El sistema valida que los datos introducidos son correctos. ✓ El sistema almacena en la base de datos los datos del nuevo tipo de licencia creado. ✓ El sistema muestra un cartel indicando que se ha adicionado correctamente el tipo de licencia y muestra el listado de todos los tipos de licencia.
		EP 1.2 Adicionar un tipo de licencia insertando algún dato incorrecto.	✓ El usuario hace la petición al sistema para adicionar un nuevo tipo de licencia. ✓ En la clase controladora se activa la función para adicionar un nuevo tipo de licencia.

			✓ Se muestra una ventana con el formulario para adicionar un nuevo tipo de licencia. ✓ El usuario entra los datos requeridos para adicionar un nuevo tipo de licencia de manera incorrecta (identificador, descripción o habilitaciones). ✓ El sistema muestra en color rojo un cartel indicando el campo con valor inválido. ✓ El usuario debe corregir el campo. ✓ El sistema valida que los datos introducidos son correctos. ✓ El sistema almacena en la base de datos los datos del nuevo tipo de licencia creado. ✓ El sistema muestra una notificación que se ha adicionado correctamente el tipo de licencia y muestra el listado de todos los tipos de licencia.
		EP 1.3 Adicionar un tipo de licencia dejando uno o más campos vacíos.	✓ El usuario hace la petición al sistema para adicionar un nuevo tipo de licencia. ✓ En la clase controladora se activa la función para adicionar un nuevo tipo de licencia. ✓ Se muestra una ventana con el formulario para adicionar un nuevo tipo de licencia. ✓ El usuario deja alguno de los campos vacíos (identificador, descripción o habilitaciones). ✓ El sistema muestra el campo vacío en rojo y muestra un cartel diciendo que el campo

			es obligatorio. ✓ El usuario debe llenar el campo vacío. ✓ El sistema valida que los datos introducidos son correctos. ✓ El sistema almacena en la base de datos los datos del nuevo tipo de licencia creado. ✓ El sistema muestra una notificación que se ha adicionado correctamente el tipo de licencia y muestra el listado de todos los tipos de licencia.
		EP 1.4 Adicionar un tipo de licencia introduciendo valores inválidos.	✓ El usuario hace la petición al sistema para adicionar un nuevo tipo de licencia. ✓ En la clase controladora se activa la función para adicionar un nuevo tipo de licencia. ✓ Se muestra una ventana con el formulario para adicionar un nuevo tipo de licencia. ✓ El usuario introduce valores inválidos (identificador, descripción o habilitaciones). ✓ El sistema muestra un cartel en rojo diciendo que el campo debe proporcionar un formato correcto. ✓ El usuario debe corregir el campo con valor incorrecto. ✓ El sistema valida que los datos introducidos son correctos. ✓ El sistema almacena en la base de datos los datos del nuevo tipo de licencia

			creado. ✓ El sistema muestra una notificación que se ha adicionado correctamente el tipo de licencia y muestra el listado de todos los tipos de licencia.

Después de aplicado el método de caja negra para validar las funcionalidades implementadas en la solución se concluye que los resultados obtenidos en cuanto a funcionalidad, fueron satisfactorios. Se les dio solución a las no conformidades detectadas durante la aplicación de las pruebas, obteniendo así un mejor funcionamiento de la aplicación.

En la Gráfico 1 se muestra el resultado de las pruebas con las no conformidades detectadas por iteración.

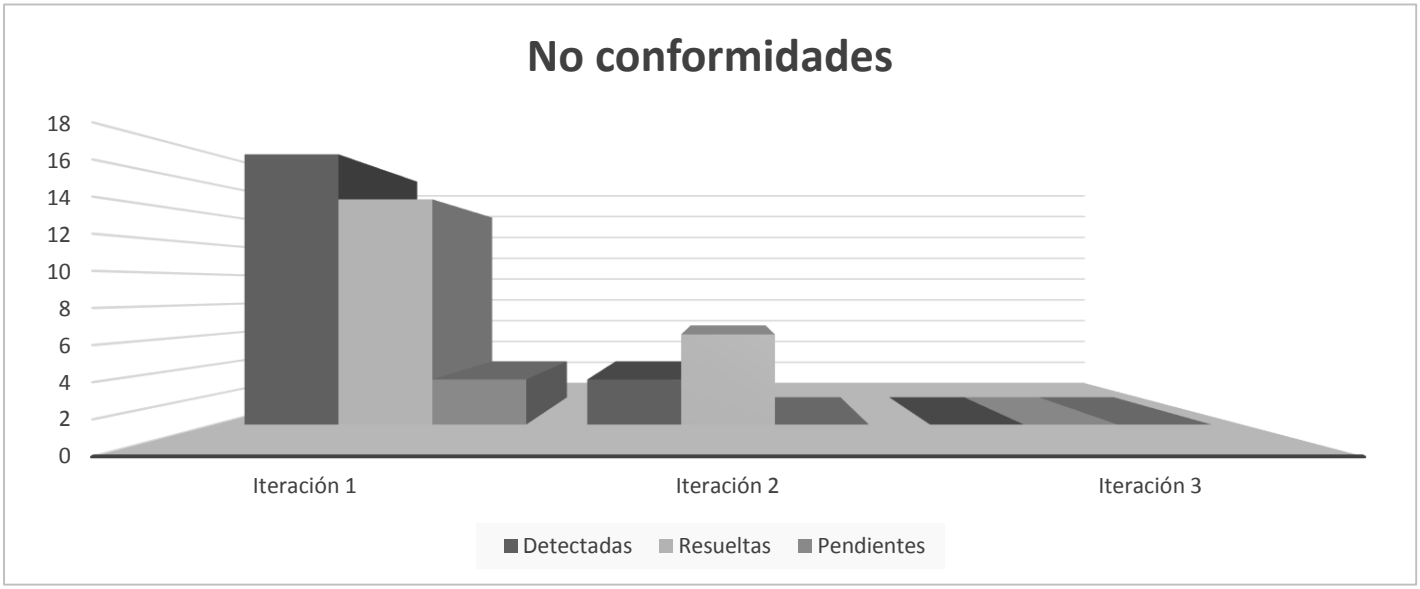


Gráfico 1. *No conformidades de la prueba de caja negra.*

(Fuente: Elaboración propia)

Para la validación de los requisitos funcionales se realizaron un total de 3 iteraciones donde se encontraron un total de 21 no conformidades, 18 en la primera iteración de las cuales 15 fueron resueltas quedando 3 pendientes. En la segunda iteración se detectaron 3 no conformidades que fueron resueltas y también las 3 pendientes de la primera iteración y en la tercera iteración no se encontraron no conformidades por lo que

la aplicación de la prueba fue satisfactoria.

3.5.5 Pruebas unitarias

Una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código. Por ejemplo, en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, se verifica que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve, que si el estado inicial es válido entonces el estado final es válido. (Barrientos, 2014)

En la aplicación de las pruebas unitarias se utiliza la técnica de **caja blanca**. Esta técnica se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Permite al ingeniero del software obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Se considera a la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican al software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. (Pressman, 2010)
Dentro del método se incluyen varias pruebas tales como:

- ✓ **La prueba del camino básico:** permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los pasos que se siguen para aplicar esta técnica son:
    1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
    2. Se calcula la complejidad ciclomática del grafo.
    3. Se determina un conjunto básico de caminos independientes.
    4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.
- ✓ **La prueba de condición:** es un método de diseño de casos de prueba que ejercita las condiciones

lógicas contenidas en el módulo de un programa.

- ✓ **La prueba de flujo de datos:** se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- ✓ **La prueba de bucles:** es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles. (Pressman, 2010)

Dentro de la técnica caja blanca se emplea el método del camino básico para realizar las pruebas. La Figura 14 muestra el código del método *editAction ()*, que permite editar una clase dado su identificador y la solicitud y en la Figura 15 se muestra el grafo de flujo asociado al método *editAction ()*.

```
(1)   public function editAction(Request $request, $id){
(1)        $em = $this->getDoctrine()->getManager();
(1)        $clase = $em->getRepository(Clase::class)->find($id);
(1)        $form = $this->createForm(ClaseType::class, $clase);
(1)        $form->handleRequest($request);
(2)        if (!$clase) {
(3)            $this->addFlash('error', 'La clase no existe.');
(3)            return $this->redirectToRoute('clase_index');
           }
(4)        if ($form->isSubmitted()) {
(5)            $nombre = $clase->getDescripcion();
(5)            $nombre = strtoupper($nombre);
(5)            $clase->setDescripcion($nombre);
(5)            $em->persist($clase);
(5)            $em->flush();
(5)            $this->addFlash('success', 'Se ha editado correctamente la clase.');
(5)            return $this->redirectToRoute('clase_index');
               }
(6)       return $this->render('AppBundle:Habilitacion\Clase:edit.html.twig', array(
(6)            'form' => $form->createView(),
(6)            'id' => $id,
              ));
       }
```

Figura 14. Método para editar una clase

(Fuente: Elaboración propia)

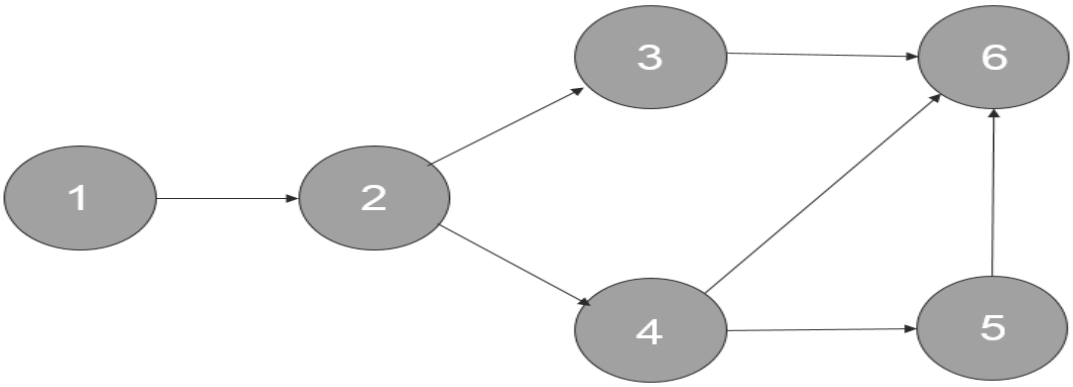


Figura 15. Grafo de flujo asociado al método editAction

(Fuente: Elaboración propia)

La complejidad ciclomática es una medición de software que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba de la ruta básica o camino básico, el valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa. Brinda una cota superior para el número de pruebas que debe realizar, a fin de asegurar que todos los enunciados se ejecutaron al menos una vez. La complejidad ciclomática tiene fundamentos en la teoría de gráficos y proporciona una medición de software extremadamente útil. La complejidad se calcula de tres formas:

1. El número de regiones del gráfico de flujo corresponde a la complejidad ciclomática.
2. La complejidad ciclomática **V(G)** para un gráfico de flujo **G** se define como:
   - **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.
   - **V(G) = P + 1** donde **P** es el número de nodos predicado[7] contenidos en el gráfico de flujo **G**. (Pressman, 2010)

En el gráfico de flujo anterior, la complejidad ciclomática es calculada usando cada uno de los algoritmos indicados:

1. El gráfico de flujo tiene tres regiones.

7 *Un nodo predicado es el que representa una condicional if o case, es decir, que de él salen varios caminos.*

2. V(G) = 7 aristas - 6 nodos + 2 = 3.
   V(G) = 2 nodos predicado + 1 = 3.

La complejidad ciclomática del gráfico de flujo asociado al método *editAction ()* tiene valor 3. V(G) proporciona la cota superior sobre el número de rutas linealmente independientes a través de la estructura de control del programa. Por tanto, las rutas o caminos posibles son las siguientes:

Tabla 8. *Rutas básicas del grafo de flujo*

(Fuente: Elaboración propia)

Número de ruta	Rutas Básicas
1	1-2-3-6
2	1-2-4-6
3	1-2-4-5-6

Una vez determinadas las rutas o caminos básicos de flujo, se pasa a ejecutar los casos de pruebas para cada camino resultante. Los datos deben elegirse de modo que las condiciones en los nodos predicados se establezcan de manera adecuada conforme se prueba cada ruta. Cada caso de prueba se ejecuta y compara con los resultados esperados. Una vez completados todos los casos de prueba, el examinador puede estar seguro de que todos los enunciados del programa se ejecutaron al menos una vez.

Tabla 9. *Caso de prueba Ruta básica 1*

(Fuente: Elaboración propia)

Ruta Básica 1: 1-2-3-6	
Descripción	Proceso para editar una clase, no es necesario introducir parámetros porque lo que se va a hacer es editar una clase existente.
Condición de ejecución	La entrada no sea una clase.
Resultado esperado	El sistema muestra el mensaje “La clase no existe”.
Resultado	Satisfactorio.

Tabla 10. *Caso de prueba Ruta básica 2*

(Fuente: Elaboración propia)

Ruta Básica 2: 1-2-4-6	
Descripción	Proceso para editar una clase
Condición de ejecución	El formulario no ha sido enviado.
Resultado esperado	El sistema redirecciona nuevamente para editar la clase.
Resultado	Satisfactorio.

Tabla 11. *Caso de prueba Ruta básica 3*

(Fuente: Elaboración propia)

Ruta Básica 3: 1-2-4-5-6	
Descripción	Proceso para editar una clase
Condición de ejecución	Proceso para editar una clase, para ello es necesario editar los atributos de la clase existente y que el formulario sea enviado.
Resultado esperado	El sistema edita la clase y guarda en la base de datos los nuevos parámetros y muestra el siguiente mensaje “Se ha editado correctamente la clase”.
Resultado	Satisfactorio.

Después de aplicado el método del camino básico, en el cual se realizaron 3 iteraciones de acuerdo a la cantidad de rutas básicas obtenidas, se concluye que, de los 3 casos de pruebas realizados, todos tuvieron resultados satisfactorios. Las no conformidades detectadas fueron resueltas, por tanto, la aplicación de la prueba fue satisfactoria en su totalidad.

3.5.6 Pruebas de seguridad

Cualquier sistema basado en computadora que gestione información sensible o cause acciones que puedan dañar (o beneficiar) de manera inadecuada a individuos es un blanco de penetración inadecuada o ilegal. La penetración abarca un amplio rango de actividades: *hackers* que intentan penetrar en los sistemas por deporte, empleados resentidos que intentan penetrar por venganza, individuos deshonestos que intentan

penetrar para obtener ganancia personal ilícita. La prueba de seguridad intenta verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia. Durante la prueba de seguridad, quien realiza la prueba juega el papel del individuo que desea penetrar al sistema. Quien realice la prueba puede intentar adquirir contraseñas por medios administrativos externos; puede atacar el sistema con software a la medida diseñado para romper cualquier defensa que se haya construido; puede abrumar al sistema, y por tanto negar el servicio a los demás; puede causar a propósito errores del sistema con la esperanza de penetrar durante la recuperación; puede navegar a través de datos inseguros para encontrar la llave de la entrada al sistema. (Pressman, 2010)

Las pruebas de seguridad se aplicaron hasta el nivel 2, el nivel 1 está dirigido a todo tipo de software y el nivel 2 es para aplicaciones que contienen datos sensibles que requieran protección. En el nivel 1 arrojó un total de 4 elementos afectados para 1 no conformidad crítica 3 no conformidades no críticas quedando evaluada de insatisfactoria la prueba. Todas las no conformidades fueron resueltas. En la segunda iteración la prueba fue evaluada de satisfactoria, debido a que no se encontraron no conformidades. A continuación, se muestran los resultados arrojados por las pruebas de seguridad de nivel 1.

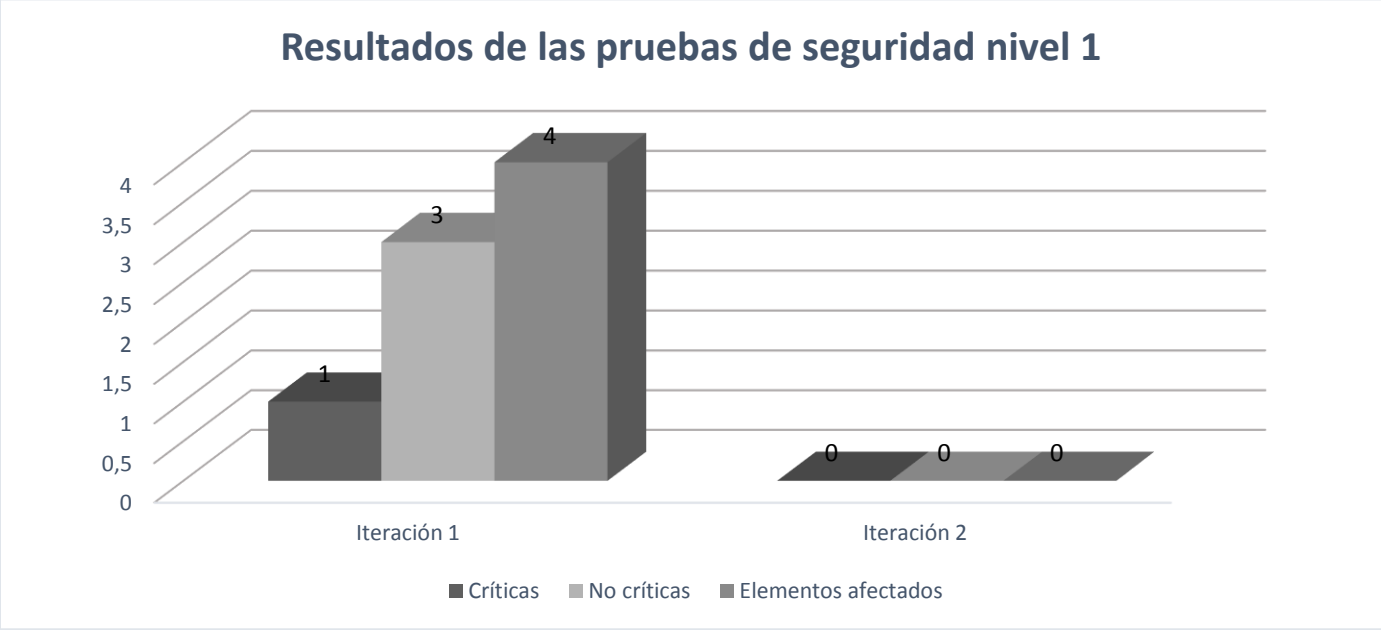


Gráfico 2. *Resultados de las pruebas de seguridad nivel 1*

(Fuente: Elaboración propia)

Se desarrollaron las pruebas de nivel 2 con la ayuda de la herramienta *Acunetix Web Vulnerabitily Scanner*[8], que establece alertas de tipo: alta, media, baja e informacional. En la primera iteración se detectaron 4 no conformidades medias, 1 no conformidad baja y 3 no conformidades informacionales para un total de 8 no conformidades, todas fueron solucionadas. Se consideran de mayor importancia las no conformidades altas y medias. Las no conformidades referentes al entorno de despliegue de la aplicación son descartadas por el equipo de desarrollo porque existen entidades encargadas de solucionarlas. A continuación, se muestran los resultados arrojados por las pruebas de seguridad de nivel 2.

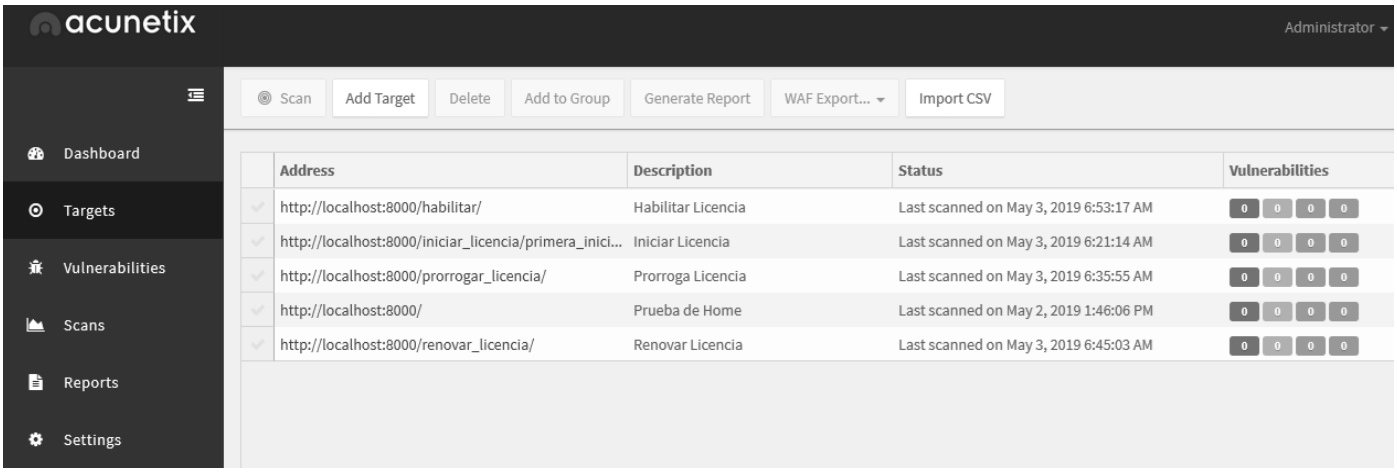


Figura 16. Resultados de las pruebas de seguridad Iteración 2 en la herramienta
(Fuente: Elaboración propia)

8 https://www.acunetix.com/vulnerability-scanner/

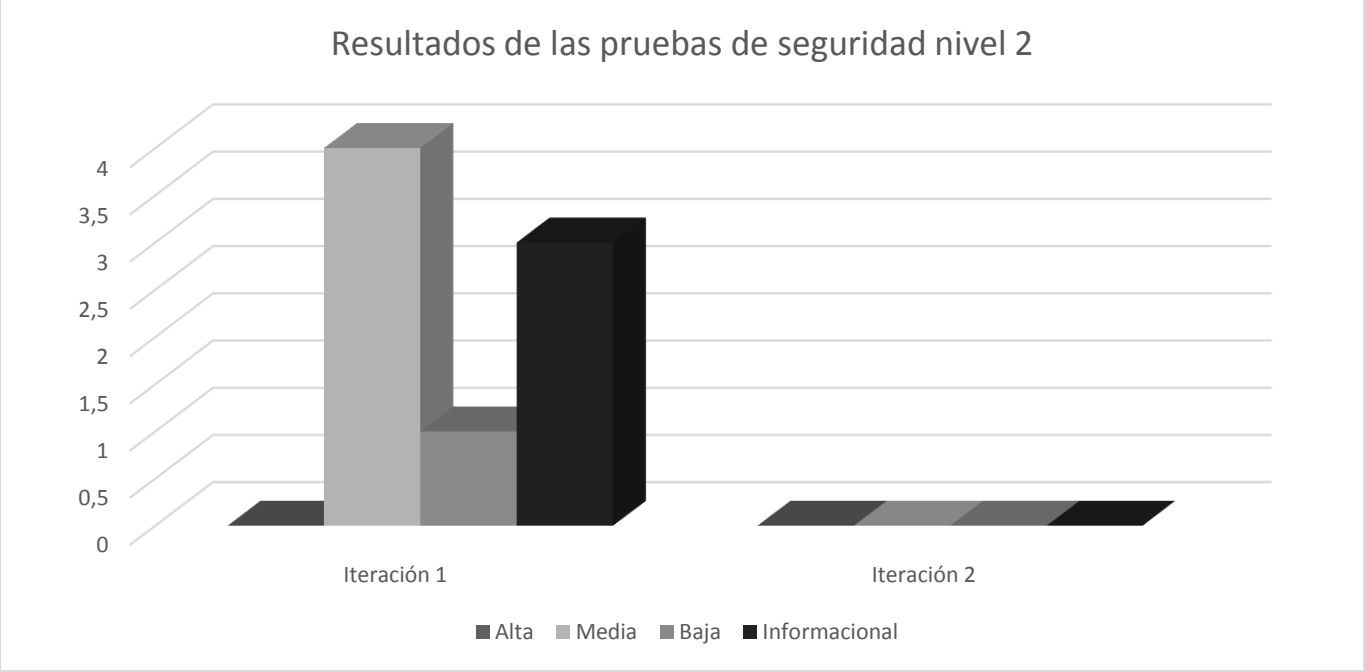


Gráfico 3. *Resultados de las pruebas de seguridad nivel 2*

(Fuente: Elaboración propia)

Conclusiones parciales del capítulo

En el capítulo concluido se elaboró el diagrama de componentes lo que permitió visualizar con facilidad la estructura general de la solución.

Se establecieron los estándares de codificación a tener en cuenta para la implementación del Sistema de Gestión de Licencias del Personal Aeronáutico del IACC versión 2.0 lo que permitió garantizar que el código posea alta calidad, reducción de errores y pueda ser mantenido fácilmente. De igual forma permite la reutilización por otros desarrolladores que lo necesiten.

Se realizaron seis tipologías de pruebas a) rendimiento, b) integración, c) aceptación, d) funcionales, e) unitarias y f) seguridad. De forma general se detectaron 33 no conformidades, todas resueltas en las diferentes iteraciones ejecutadas.

Conclusiones generales

- ✓ La caracterización de los principales sistemas informatizados para la gestión de los procesos de gestión y emisión de licencias para el personal de la aeronáutica civil arrojó, que están desarrollados de forma personalizada para cada institución a partir de las regulaciones sobre seguridad e integridad de la gestión de documentación oficial emitidas por la OACI.
- ✓ La caracterización de las regulaciones emitidas posibilitó entender los principales procesos, módulos y funcionalidades que estos deben cumplir.
- ✓ La metodología AUP-UCI escenario 4, permitió la obtención de un conjunto de métodos coherentes y relacionados por principios comunes en el proceso de desarrollo del Sistema de Gestión de Licencias del Personal Aeronáutico versión 2.0.
- ✓ Se obtuvieron 58 requisitos funcionales y 8 no funcionales a partir del proceso de identificación de los requisitos y los artefactos generados. Estos constituyeron las capacidades y las cualidades de la propuesta de solución respectivamente.
- ✓ La actualización de la versión de Symfony a la versión 3.4.28 permitió la corrección de los problemas de seguridad detectados.
- ✓ La optimización de las consultas permitió mejorar el rendimiento del sistema y de esta forma la capacidad de respuesta, el servicio que se ofrece y la satisfacción de los usuarios.
- ✓ La realización nuevos reportes para el apoyo a la toma de decisiones informada permitieron mejorar la supervisión y el control por parte del personal administrativo.
- ✓ La realización de las pruebas de software seleccionadas permitieron la verificación de la propuesta de solución.

Recomendaciones

- ✓ Usar el servicio de identificación de los ciudadanos del Ministerio del Interior.
- ✓ Creación de un editor de plantillas de licencias de vuelos.

Referencias

Aerocivil. (11 de octubre de 2018). Recuperado el 19 de enero de 2019, de http://www.aerocivil.gov.co/autoridad-de-la-aviacion-civil/certificacion-y-licenciamiento/Licencias%20a%20personal%20aeronutico/SISTEMA%20INFORMACI%C3%93N%20GESTI%C3%93N%20AERON%C3%81UTICA%20-%20SIGA%20(1).pdf

Arias, M. C. (19 de marzo de 2019). Obtenido de http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf

Barrientos, P. A. (2014). *Enfoque para pruebas de unidad basado en la generación aleatoria de objetos.*

DGAC. (19 de enero de 2019). *Nuevo sistema de licencias y habilitaciones aeronauticas*. Recuperado el 20 de marzo de 2019, de https://www.dgac.gob.cl/nuevo-sistema-de-licencias-y-habilitaciones-aeronauticas-dgac/

Eguiluz, J. 2. (2018). *Capítulo 1. Introducción (Introducción a CSS)*. Obtenido de https://librosweb.es/libro/css/capitulo_1.html

Gómez, C., & Olive, A. (2003). *Diseño de sistemas software en UML.* Barcelona, España: EDICIONS UPC.

Group, T. P. (18 de marzo de 2019). *PHP: ¿Qué es PHP? - Manual*. Recuperado el 21 de 03 de 2019, de http://php.net/manual/es/intro-whatis.php

IBM. (15 de diciembre de 2004). *The component diagram*. Recuperado el 14 de abril de 2019, de https://www.ibm.com/developerworks/rational/library/dec04/bell

INAC. (20 de abril de 2019). *Instituto Nacional de Aeronáutica Civil*. Recuperado el 02 de mayo de 2019, de http://www.inac.gob.ve/

Instituto de Aeronáutica Civil de Cuba. (13 de octubre de 2017). *Portal del Instituto de Aeronáutica Civil de Cuba.* Recuperado el 4 de febrero de 2019, de http://www.iacc.gob.cu/

Iturralde, O. J. (2016). *Introducción a Los Patrones de Diseño: Un Enfoque Práctico.* Platform, CreateSpace Independent Publishing.

Izaurralde, M. P. (2013). Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario. Universidad Tecnológica Nacional , Córdova.

JetBrains. (2019). *PhpStorm*. Obtenido de https://www.jetbrains.com/phpstorm/?fromMenu

Jiménez, J. L. (13 de 06 de 2016). *UF2406 - El cliclo de vida del desarrollo de aplicaciones.* Elearning, S.L.

Krall, C. (2018). *¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML.* Recuperado el 19 de 3 de 2019, de https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-

es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163

León, S. T. (2018). *UF2213 - Modelos de datos y visión conceptual de una base de datos.* Elearning, S.L.

PostgresSQL. (21 de 4 de 2019). *Características, limitaciones y ventajas*. Recuperado el 21 de 3 de 2019, de http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html

Pressman, R. S. (2010). *INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO.* México: McGRAW-HILL. Obtenido de www.FreeLibros.me

Salmeron, J. M. (2019). *Misión - Visión - Valores - Slogan | INAC*. Obtenido de http://www.inac.gob.ve/?page_id=1405

SensioLabs. (2019). En SensioLabs, *Symfony 3.4 The Best Practices Book* (págs. 6-11). Obtenido de https://symfony.com/pdf/Symfony_best_practices_3.4.pdf

SIPA. (20 de diciembre de 2018). *Dirección General de Aeronáutica Civil - Sistema Informático de Personal Aeronáutico*. Recuperado el 23 de abril de 2019, de https://sipa.dgac.gob.cl/

Sommerville, I. (2005). Madrid: Pearson Educación.

Symfony. (2019). *¿Qué es Symfony?* Recuperado el 18 de marzo de 2019, de https://symfony.es/pagina/que-es-symfony

Symfony. (18 de enero de 2019). *Twig*. Recuperado el 29 de enero de 2019, de https://twig.symfony.com/

Tecnología e Innovación. (2015). *Vol.2* (5 980-986). Obtenido de https://ecorfan.org/bolivia/researchjournals/Tecnologia_e_innovacion/vol2num5/Tecnologia_e_Innovacion_Vol2_Num5_6.pdf

UCI. (2018). Obtenido de https://publicaciones.uci.cu/?journal=SC&page=article&op=viewFile&path[]=1840&path[]=937

Universidad de las Ciencias Informáticas. (octubre de 2018). *Portal de la Universidad de las Ciencias Informáticas*. Recuperado el 14 de diciembre de 2018, de https://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo/centro-de-identificacion-y-seguridad-digital-cised

Uniwebsidad. (noviembre de 2018). *2.1. El patrón MVC (Symfony 1.4, la guía definitiva)*. (uniwebsidad) Obtenido de https://uniwebsidad.com/libros/symfony-1-4/capitulo-2/el-patron-mvc

Visual Paradigm. (diciembre de 2018). *Solución de diagramas y software de diagramas en línea*. Recuperado el 15 de marzo de 2019, de https://online.visual-paradigm.com/es/

Web, M. d. (07 de 03 de 2007). *¿Qué es Javascript?* Recuperado el 12 de 3 de 2019, de http://www.maestrosdelweb.com/que-es-javascript

SISTEMA DE GESTIÓN DE LICENCIAS

Anexos

Anexo 1. Modelo de datos

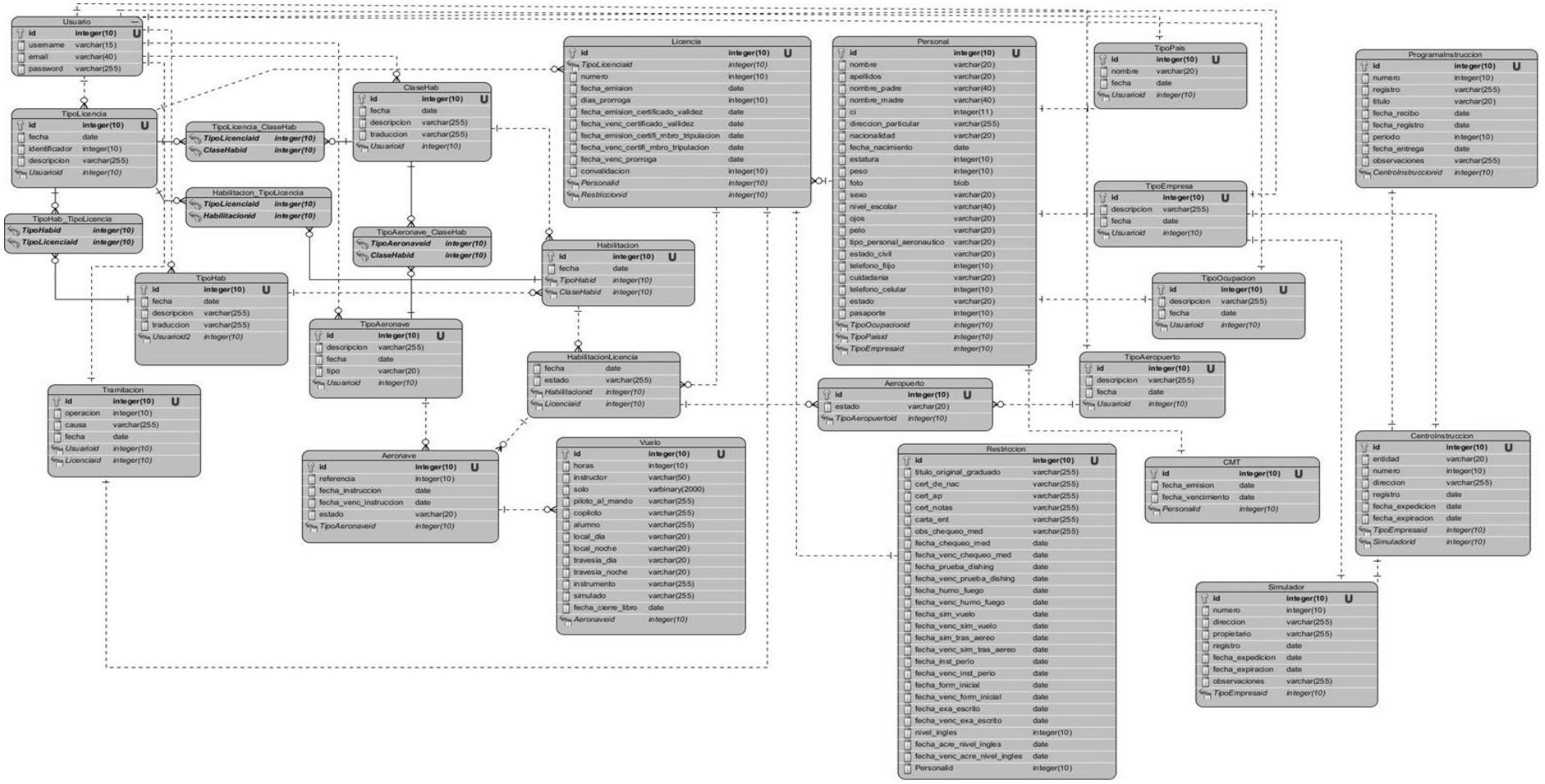


Figura 17. Modelo de datos

(Fuente: Elaboración propia)

SISTEMA DE GESTIÓN DE LICENCIAS

Anexo 2. Diagrama de clases

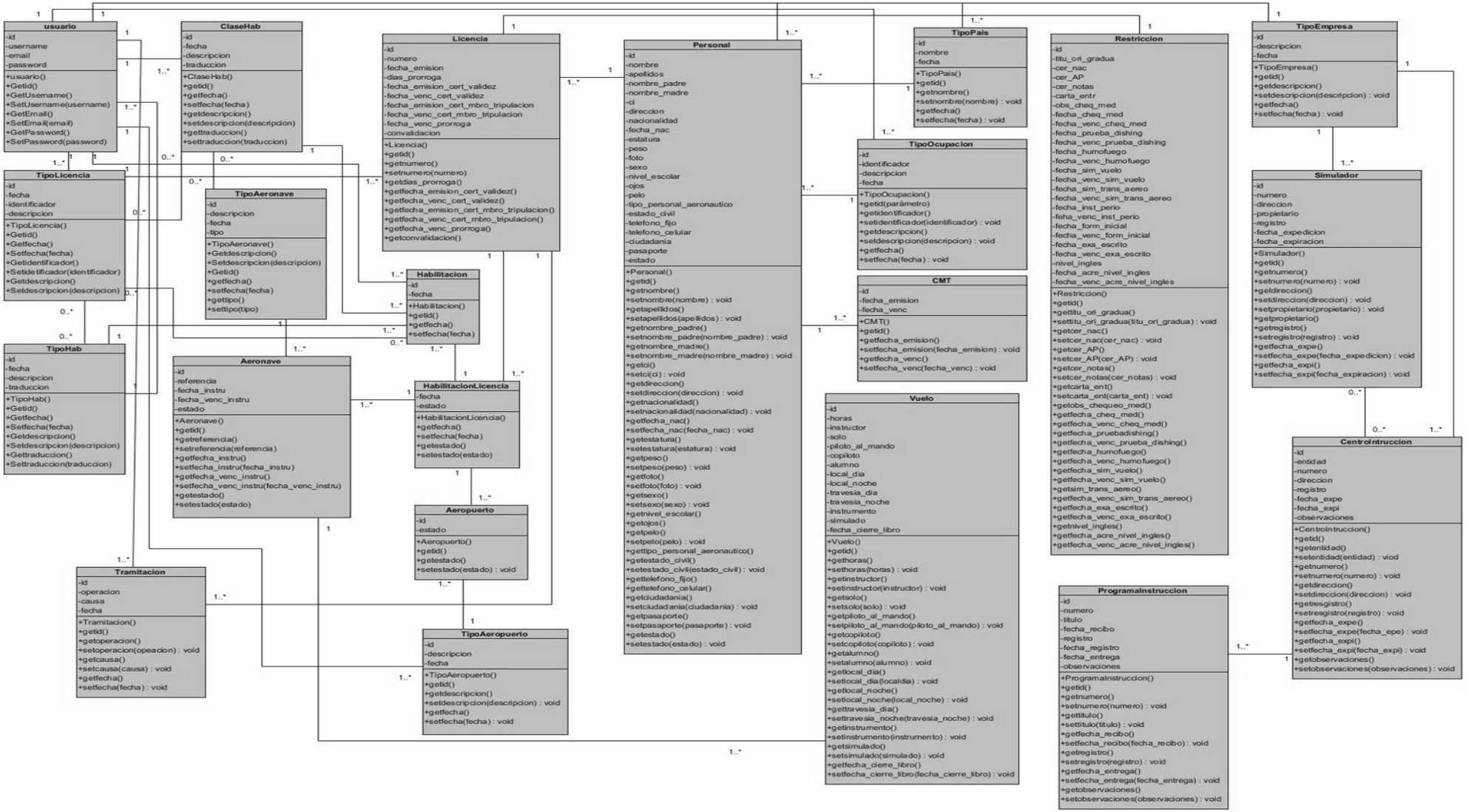


Figura 18. Diagrama de clases

(Fuente: Elaboración propia)

Anexo 3. Historia de Usuario

Tabla 12. *Historia de usuario: Eliminar habilitación*

(Fuente: Elaboración propia)

Historia de Usuario	
Nombre: Eliminar habilitación	**Número:** HU_5
Modificaciones: Ninguna	
Programador: Yunier Pérez Bejerano	**Iteración Asignada:** 1
Prioridad en el negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Bajo	**Puntos Reales:**
Descripción: Se permitirá eliminar habilitaciones al sistema.	
Observaciones:	
Prototipo de interfaz de usuario:	

Tabla 13. *Historia de usuario: Realizar iniciación de licencia piloto*

(Fuente: Elaboración propia)

Historia de Usuario	
Nombre: Realizar iniciación de licencia piloto	**Número:** HU_6
Modificaciones: Ninguna	
Programador: Yunier Pérez Bejerano	**Iteración Asignada:** 2
Prioridad en el negocio: Media	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos Reales:**
Descripción: Se permitirá: • Añadir restricciones • Elegir tipo de licencia (Alumno Piloto (AP), Piloto privado (PP), Piloto comercial (PC), Piloto con tripulación múltiple (MPL) – avión, Piloto de Transporte de Línea Aérea (PTLA)) • Buscar el personal. • Chequear restricciones de acuerdo al tipo de licencia. Restricciones generales a comprobar: o Titulo Original de Graduado (SI, NO)	

- Certificación de nacimiento (SI, NO)
- Certificación de antecedentes penales (SI, NO)
- Certificado de notas (SI, NO)
- Carta de solicitud de la entidad donde labora (SI, NO)
- Observaciones referentes al chequeo médico (Texto) (Protectores auditivos, Uso lentes correctores)
- Fecha del chequeo médico (Fecha(dd/mm/aa))
- Fecha de vencimiento del chequeo médico(Fecha(dd/mm/aa))
- Fecha de la prueba del dishing (Fecha(dd/mm/aa))
- Fecha de vencimiento de la prueba del dishing (Fecha(dd/mm/aa))
- Fecha de la prueba del humo y fuego (Fecha(dd/mm/aa))
- Fecha de vencimiento de la prueba del humo y fuego (Fecha(dd/mm/aa))
- Fecha del simulador de vuelo (Fecha(dd/mm/aa))
- Fecha de vencimiento del simulador de vuelo (Fecha(dd/mm/aa))
- Fecha de la instrucción (Fecha(dd/mm/aaaa))
- Fecha de vencimiento de la instrucción (Fecha(dd/mm/aa))
- Fecha de la formación inicial (Fecha(dd/mm/aaaa))
- Fecha de vencimiento de la formación inicial (Fecha(dd/mm/aa))
- Nivel de inglés (Valor numérico en el rango de [4-6])
- Fecha de acreditación del nivel de inglés (Fecha(dd/mm/aa))
- Fecha de vencimiento de la acreditación del idioma inglés (Fecha(dd/mm/aa))

- Registrar:
  - Número de licencia (Número de 4 cifras que se genera por el sistema).
  - Fecha en que se emitió (dd/mm/aa).
  - Fecha en que se vence certificado miembro tripulación (dd/mm/aa).
  - Fecha en que se vence certificado de validez de la licencia (dd/mm/aa).
  - Días de prórroga.
  - Fecha de vencimiento con prórroga.
- Agregar habilitaciones.
- Generar licencia (Licencia, Certificado de validez, Certificado de miembro Tripulación)

Observaciones:

Prototipo de interfaz de usuario: