Universidad de las Ciencias Informáticas

Facultad 3

Módulo Guardia Obrera Estudiantil versión 2.0 para el Sistema de Administración y Economía de la Facultad 3

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

Autor:
Osvaldo Alejandro Pacheco Morales

Tutora:
MSc. Ana Marys Garcia Rodríguez

Co-tutor:
Ing. Juan Darién Macías Hernández

La Habana, Junio de 2016

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autor:

_____

Osvaldo Alejandro Pacheco Morales

Tutora:

_____

MSc. Ana Marys Garcia Rodríguez

Co-tutor:

_____

Ing. Juan Darién Macías Hernández

DATOS DE CONTACTO

Síntesis de la Tutora:
Nombres y apellidos: Ana Marys Garcia Rodríguez.
Correo: agarcia@uci.cu

Profesora Asistente de Ingeniería y Gestión de Software.
Ingeniera en Ciencias Informáticas.
Máster en Calidad de Software.

Síntesis del Co-tutor:
Nombres y apellidos: Juan Darién Macías Hernández.
Correo: macias@uci.cu

Ingeniero en Ciencias Informáticas.

AGRADECIMIENTOS

Deseo agradecer ante todo a suprema tutora, Ana Marys y co-tutor Macías, por todo el esfuerzo y dedicación que me prestaron; noches en vela, cansancio y esfuerzos compartidos para poder llegar a donde estoy; les agradezco ahora y nunca será suficiente. A Alejandro que fue mi otro pilar en la lucha que sostenía, siempre que necesitaba su ayuda ahí estaba para brindármela, mejor dicho para sacarme del apuro. Agradecerle a Javier que ha sido el compañero con el cual he recorrido este largo camino, en las buenas y en las malas, literalmente. Agradecer a todas mis amistades, las que están presentes y las que no, empezando por piquete del dominó y café: Yordan, Falcón, Ever, Boris, Eimé. A aquellos que nunca dejaron de insistirme y preocuparse para que saliera adelante: Romilio, Frank, Yasmany a Yerandi y Luis Guillermo quienes, a pesar de no estar, me escribían constantemente preguntado cómo me iba, que avances tenía. A los Yolo, Victor y Alejandro por los buenos momentos que pasamos y por la ayuda prestada. Agradecer sinceramente a la oponente y al tribunal por tener tantas concesiones y solidarizarse con la situación en la que estaba, por la ayuda prestada. En fin gracias a todos por venir y por compartir este momento especial de mi vida.

DEDICATORIA

Deseo dedicar la presente investigación a mis padres, que son mi guía y modelo a seguir, que constituyen la razón por la que he trabajado durante todo este tiempo para llegar a este momento. Para ellos, que en los momentos que toqué fondo, me alentaron y apoyaron, que me dieron razones a continuar y no desistir. Hoy les digo: mi éxito es su éxito, disfrútenlo, padézcanlo, que les pertenece a ustedes.

Dedicárselo también a mi familia, mi abuela Cuca, mi tío Roge, mis primas, Daniela y Denisse, a mis primos Angelito y Javier, mi abuelo Mario y mi tío Ángel, siempre los llevo presentes.

RESUMEN

La Universidad de las Ciencias Informáticas constituye un programa desarrollado con la misión de formar profesionales competentes y comprometidos con la Revolución. Entre los procesos de apoyo que desarrolla este centro de altos estudios, se encuentra la Guardia Obrera Estudiantil, que es planificada y controlada por cada área. El Vicedecanato de Economía y Administración de la Facultad 3, desarrolló un sistema para el control de los procesos asociados a la planificación y control de la Guardia Obrera Estudiantil en el año 2015, sin embargo, se identificaron en el área oportunidades de mejora a implementar para mejorar el control de la información de dichos procesos. La investigación persigue como objetivo el desarrollo de la versión 2.0 del módulo Guardia Obrera Estudiantil para el Sistema de Administración y Economía de la Facultad 3. Para el desarrollo de la investigación se realiza un estudio de la metodología a aplicar como guía del proceso de desarrollo de software y se definen las principales herramientas y tecnologías empleadas en la implementación de la solución; además, se describe la arquitectura que da soporte a la solución. El diseño, la implementación y la solución propuesta fueron validados a través de las métricas del diseño, las pruebas de caja negra y la técnica de Iadov.

Palabras claves: control, control de información, Guardia Obrera Estudiantil, planificación.

ÍNDICE

INTRODUCCIÓN

Una parte clave en toda organización es la protección de los activos fijos de la empresa contra las amenazas que se vierten en la sociedad -ya sea la malversación o el robo de la propiedad estatal y privada- a las que constantemente se ven sometidas las organizaciones que manejan recursos materiales. La Universidad de Ciencias Informáticas (UCI) no se encuentra exenta a ello, por lo que una de las actividades desarrolladas como parte de la seguridad y protección, es la planificación y control de la Guardia Obrera Estudiantil (GOE).

Dada la importancia de esta actividad, así como la complejidad de los procesos involucrados y los altos niveles de información requeridos, se evidenció la necesidad de la informatización de los procesos asociados a la planificación y control de la GOE en la Facultad 3, razón por la cual es desarrollada una primera versión del actual Sistema de Administración y Economía de la Facultad 3 (SAEF3) en el año 2015. El módulo GOE del SAEF3, se encarga de la gestión de los procesos asociados a la planificación y control del GOE, sin embargo, se han identificado oportunidades de mejoras mediante la modificación de funcionalidades ya existentes y la implementación de nuevas funcionalidades.

Para mejorar el control sobre el cumplimiento de la GOE, se consideró oportuno la modificación de las funcionalidades asociadas a:

- Configuración de la información básica para el desarrollo de la GOE.
- Configuración de las formas de rotaciones por turnos ya que pueden variar indistintamente.
- Restricción de las fechas de cumpleaños para el personal involucrado. Insertar otro conjunto de restricciones referentes a fechas, afinidad y días de la semana de manera que la planificación automática sea más versátil, para poder así humanizar el proceso teniendo en cuenta las necesidades que puede presentar el personal involucrado en la planificación.
- Establecimiento de restricciones en las permutas de acuerdo a las pautas inviolables; no dar la posibilidad de que una persona realice más de una solicitud de permuta a la vez y permitir la cancelación de una solicitud de permuta.
- Análisis previos a la planificación a partir del potencial disponible, para visualizar la factibilidad de la configuración definida, brindar la posibilidad de ajustarla antes de realizar la planificación y así evitar inconformidades con el resultado final.
- Visualización por áreas, personas y rangos de fechas, del histórico de guardias planificadas

para facilitar la búsqueda de información.

Además fue necesaria la inclusión de nuevas funcionalidades asociadas a:

- Registro del cumplimiento de la planificación de GOE para facilitar el control sobre el proceso.
- Visualización de los incumplimientos en rangos de fechas específicos por los jefes de áreas, como parte del nivel de información que brinda el módulo GOE.
- Configuración de notificaciones para personalizar los mensajes según la acción que se ejecuta.
- Notificación a los implicados en los incumplimientos, así como a los jefes de áreas para elevar el nivel información brindado a los usuarios.
- Generación del parte del cumplimiento de la GOE posibilitando el filtrado por un rango de fecha determinado.
- Establecer la autenticación (por usuarios UCI) mediante el consumo de servicios, así como la gestión del potencial empleando dichos servicios, a fin de lograr que el módulo sea aplicable a otros escenarios de la universidad.

A partir de lo antes expuesto se define como problema a resolver: ¿cómo elevar el control de la información asociada a los procesos de planificación y control de la GOE en la Facultad 3?

Se identifica como objeto de estudio: procesos de planificación y control de la GOE; enmarcado en el campo de acción: informatización de los procesos de planificación y control de la GOE en la Facultad 3.

Se plantea como idea a defender: el desarrollo de la versión 2.0 del módulo GOE para el SAEF3, contribuirá a elevar el control de la información asociada a los procesos de planificación y control de la GOE en la Facultad 3.

Para dar solución al problema planteado se traza como objetivo general: desarrollar la versión 2.0 del módulo GOE para el SAEF3, de manera que contribuya a elevar el control de la información asociada a los procesos de planificación y control de la GOE en la Facultad 3. El objetivo general se desglosa en los siguientes objetivos específicos:

1. Definir el marco teórico de la investigación mediante el estudio y el análisis de los principales referentes teóricos para el desarrollo de la solución.

2. Realizar el diseño e implementación de la solución para obtener los componentes de software de los procesos asociados a la planificación y control de la GOE en la Facultad 3.
3. Valorar la efectividad de la solución propuesta mediante la realización de pruebas de caja negra y la aplicación de la técnica Iadov.

Para lograr el cumplimiento de los objetivos trazados se aplicaron los métodos de investigación:

Métodos teóricos:

- Análisis histórico-lógico: empleado para la recopilación de datos históricos sobre el desarrollo de sistemas para la planificación y control de la GOE.
- Analítico-sintético: para el análisis y la comprensión de la documentación, lo cual permitió realizar un estudio teórico de la investigación mediante la comparación de diferentes sistemas, así como precisar los elementos y características del diseño de la propuesta de solución.
- Modelación: para el análisis y diseño del módulo GOE del SAEF3 a través de las tecnologías y herramientas seleccionadas.

Métodos empíricos:

- Entrevista: para aplicar entrevistas no estructuradas a trabajadores del VDEA de la Facultad 3, y así obtener información necesaria sobre el funcionamiento del negocio.
- Observación: para obtener información del entorno a partir de una percepción propia, se realiza durante toda la investigación como método de esclarecimiento.

Cualitativos y cuantitativos:

- Los modelos de análisis cuantitativos para el procesamiento de los resultados de las encuestas y entrevistas realizadas.

Se aplicó además la técnica Iadov para conocer el nivel de satisfacción de los clientes con la solución propuesta.

El trabajo de diploma se encuentra estructurado en tres capítulos de la siguiente manera:

Capítulo 1: se analizan los referentes teóricos asociados al objeto de la investigación. Se estudian los

principales conceptos que se relacionan con la investigación para una mejor comprensión del negocio, lo cual incluye el análisis de sistemas homólogos a nivel nacional e internacional para la gestión de la GOE. Se realiza una descripción de la metodología, herramientas y tecnologías seleccionadas para el desarrollo de la solución.

Capítulo 2: se describen los nuevos requisitos del sistema aplicando las técnicas de captura y validación de requisitos. Se describe el diseño de la solución especificando los patrones de diseño utilizados y se muestran los resultados alcanzados mediante el uso de las métricas del diseño para verificar el correcto uso de los patrones del diseño.

Capítulo 3: se describen los componentes y clases generadas durante la implementación del módulo, así como de los estándares de codificación empleados. Se especifican los resultados de las pruebas aplicadas para la verificación del módulo, así como el grado de satisfacción de los clientes con la solución propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se definen los principales conceptos relacionados con la investigación y se realiza un estudio de los sistemas homólogos. Se describe la metodología seleccionada para el desarrollo, así como de las herramientas y tecnologías para la implementación de la solución.

1.2 Conceptos asociados al dominio del problema

Gestión de la información: constituye un proceso mediante el cual se planifican, organizan, dirigen y controlan los recursos de información de una organización asegurando un adecuado tratamiento, intercambio y uso de este recurso, de manera que contribuyan al establecimiento de fortalezas organizacionales, permitiendo que todos sus miembros dispongan de forma adecuada de los recursos informativos que existen en ella (Orozco et al. 2009).

Planificación: plan general, metódicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo determinado (RAE 2014). Constituye el establecimiento de objetivos y metas, una actividad continua y unitaria que no termina con la formulación de un plan determinado, sino que implica un reajuste permanente entre medios, actividades, fines, caminos, procedimientos y la elección de los medios más convenientes para alcanzarlos como elementos comunes e importantes para definir la planificación (Pedraza 2009).

Control: conjunto de actividades que se emprenden, para medir, examinar y evaluar la ejecución de los planes y los resultados obtenidos, con el fin de detectar y prever desviaciones, diagnosticando la razón de las desviaciones y tomando las medidas correctivas necesarias para asegurar la obtención de los objetivos (Pañeda 2004).

1.3 Análisis de soluciones existentes

1.3.1 Sistema Web de Asignación Automática de Mallas de Turnos para Protector Security (AutoMaT P.S)

Sistema web online que asigna automáticamente los turnos para Protector Security, una empresa que presta servicios de seguridad y vigilancia en diversas ciudades de Chile. El sistema maneja distintos

niveles de usuarios y genera reportes mensuales; de esta manera, los supervisores no tienen la facultad de asignar los horarios de trabajo. Se evitan pérdidas económicas por turnos inexistentes, generación de errores, ineficiencia y pérdida de recursos. El producto está disponible 7 días x 24 horas en condiciones normales y contempla un uso óptimo de las conexiones a la base de datos. Es una aplicación que requiere de buenas prestaciones a INTERNET, así como requerimiento de hardware máximo, aunque presenta una escasa documentación técnica (Gutiérrez and Tapia 2013).

1.3.2 Jano Seguridad (JANO)

Es un sistema de información desarrollado para empresas y organizaciones que prestan servicios de vigilancia, seguridad y control. Incorpora un modelo completo de elaboración de cuadrantes de servicios, sean puestos fijos o personal de ronda. Incluye la gestión administrativa del servicio, incidencias, costes y rentabilidad. Facilita el tratamiento de las condiciones comerciales de cada cliente, e integra en un mismo entorno la gestión corporativa. Incluye la gestión de ofertas y presupuestos especiales del sector, así como la gestión administrativa del servicio, incidencias, costes y factibilidad. Facilita el tratamiento de las condiciones comerciales de cada cliente, e integra en un mismo entorno la gestión corporativa (EMR 2008).

1.3.3 Aplicación web para la gestión de la planificación de la GOE en la UCI (PGOE-UCI)

Es un sistema informático desarrollado para la gestión de la planificación de la GOE en la UCI. Identifica una serie de reglas a tener en cuenta que pueden dificultar o complejizar la planificación cuando es ejecutada por una persona, logrando que no se repitan consecutivamente los días, turnos y postas de guardia y se satisfagan las necesidades básicas para el cumplimiento de la misma. Es independiente del sistema operativo donde se ejecute y presenta un requerimiento de hardware mínimo (Jiménez and Concepción 2012).

1.3.4 Sistema de gestión docente metodológica integrado con la Plataforma de Servicios Integrados para la Facultad 3 (SisGes-DM)

Aplicación informática que dio solución a las diferentes deficiencias concernientes a los procesos: Plan de trabajo metodológico, Guardia docente y Control a clases. Permite la centralización, así como una adecuada organización y una eficiente gestión de la información. Además, provee una retroalimentación de datos entre los involucrados, generación de estadísticas, una mayor disponibilidad

y actualización de la información. Es independiente del sistema operativo donde se ejecute y presenta un requerimiento de hardware mínimo. Es capaz de centralizar y gestionar la información de los procesos docentes metodológicos y de guardia docente adscritos a la Facultad 3 de la UCI (Campos and Méndez 2013).

1.3.5 Sistema para la Gestión de la Guardia Obrera-Estudiantil en la Facultad 3 (SisGest GOE-Fac3)

Es un sistema de gestión para la planificación ágil y más organizada de la Guardia Obrera-Estudiantil, permite la gestión de procesos que tributan a un adecuado desempeño, organización y funcionamiento, facilitando el manejo y gestión de la información de dicha actividad con una mayor rapidez en su ejecución. Brinda la posibilidad de gestionar la planificación de la guardia en la Facultad 3, a través de una serie de requisitos y restricciones. Realiza cambios o permutas de guardias entre estudiantes, siempre notificando a través del correo. Además se registran las incidencias ocurridas en el transcurso de la guardia cada día (Mata and Navarro 2013).

1.3.6 Aplicación Web para el control de la Guardia Estudiantil y la cuartelería de la Facultad 7 (Bejerano and Montes de Oca 2009)

Es una aplicación web que visualiza la gestión de la información y planificación de la guardia estudiantil. Persigue el objetivo de proporcionarle a la dirección de la facultad la posibilidad de tener registrada toda la información necesaria referente a los procesos de la guardia estudiantil. Lo que posibilita realizar de forma más eficiente el manejo y gestión de la información de estas actividades. Además, proporciona un control personalizado de la evaluación de los estudiantes en la residencia estudiantil y la generación de un conjunto de reportes en formato digital. Es independiente del sistema operativo donde se ejecute y presenta un requerimiento de hardware mínimo (Bejerano and Montes de Oca 2009).

1.3.7 Módulo GOE del SAEF3

Es un módulo del SAEF3, destinado a la planificación y control de la GOE. En su primera versión dio solución a necesidades relacionadas con la planificación, intercambio de guardias planificadas, visualización de la planificación, visualización de la factibilidad y posibilidad del control de la misma mediante la generación de información asociada a la planificación. Las funcionalidades implementadas

facilitan en gran medida el esfuerzo y tiempo de realizar las actividades de manera manual (Aguilar and Castillo 2015).

1.3.8 Resultados del análisis de sistemas homólogos

Se realizó el análisis de los sistemas homólogos considerando además, la medida en que los mismos satisfacen las necesidades de informatización de los procesos de planificación y control de la GOE que se desarrollan dentro del VDEA en la Facultad 3. En la siguiente tabla comparativa se muestran las necesidades suplidas por cada uno de los sistemas estudiados.

Tabla 1: Tabla comparativa de los sistemas homólogos. Fuente: (Elaboración propia).

	AutoMaTP.S	JANO	CGEC-Fac7	PGOE-UCI	SisGes-DM	SisGest GOE-Fac3	GOE-SAEF3
Planificar guardia	x	x	x	x	x	x	x
Registrar cumplimiento	-	-	-	-	x	-	-
Realizar permutas entre implicados	-	-	-	-	-	-	-
Gestionar restricciones al personal implicado	-	-	x	-	-	-	x
Análisis de factibilidad	-	x	-	-	-	-	x
Generación de información relacionada a la planificación	x	x	x	x	x	x	x
Intercambiar planificación	-	-	-	x	-	x	x

Tras el análisis de las alternativas brindadas para resolver el problema asociado a la planificación y control de la GOE, se pudo constatar que a pesar de las ventajas brindadas por cada uno de los

sistemas, en el caso de las soluciones desarrolladas fuera del entorno de la UCI, no permiten el control del cumplimiento de la guardia. En todos los casos estudiados, a excepción del SAEF3, no se realiza la gestión de restricciones en el personal para la planificación de la guardia, no se configuran previamente las postas, turnos y personal adecuado a cubrir en cada turno-posta. A partir del análisis anterior se identifica como una solución más completa el módulo GOE del SAEF3; sin embargo, se identificaron oportunidades de mejoras asociadas a que: no se brinda la posibilidad de permuta de turnos entre el personal, no permite visualizar previo a la planificación, la efectividad de configuración de la guardia diseñada, no realiza notificaciones de los incumplimientos a los jefes de área, no realiza una configuración efectiva de la rotación de los turnos y no permite gestionar las restricciones. Por lo anterior, se concluye que las soluciones analizadas no satisfacen completamente las necesidades particulares de los procesos de planificación y control de la GOE de la Facultad 3, aunque constituyen aportes a la solución, la forma de manejar y mostrar la información de los sistemas precedentes a fin de obtener una solución que satisfaga el problema planteado.

1.4 Proceso de desarrollo de software

Un proceso de desarrollo de software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto. Este conjunto de actividades en el proceso de desarrollo de software tiene la misión de transformar los requerimientos del usuario en un producto de software; de manera que los integrantes del equipo y todo aquel interesado en el producto final, tengan la misma visión (Jacobson et al. 2000).

1.4.1 Metodología de desarrollo

Una metodología consiste en múltiples herramientas, modelos y métodos para asistir el proceso de desarrollo de software, donde se definen con precisión los artefactos, actividades y roles involucrados, así como las prácticas y técnicas recomendadas, las guías de adaptación de la metodología al proyecto y las guías para el empleo de herramientas de apoyo (Figueroa et al. 2008).

En el contexto de la investigación resulta esencial la aplicación de una metodología con enfoque ágil, debido al reducido número de miembros del equipo de desarrollo, la necesidad de implementación de la solución en un período de tiempo corto y la posibilidad de contar con el cliente como parte del

equipo de desarrollo y no a tiempo parcial. Entre las metodologías ágiles destaca por su adaptación el Proceso Unificado Ágil (AUP por sus siglas en inglés).

AUP

El Proceso Unificado Ágil de Scott Ambler, es una versión simplificada del Proceso Unificado de Rational (RUP). Describe de una manera simple y sencilla, la forma de desarrollar aplicaciones de software de negocio empleando técnicas ágiles y conceptos que aún se mantienen efectivos en RUP. Entre las técnicas ágiles que aplica incluye (Ambler 2009):

- Desarrollo dirigido por pruebas
- Modelado ágil
- Gestión de cambios
- Refactorización de base de datos para mejorar la productividad

Metodología Variación AUP-UCI

Para el desarrollo de la solución se seleccionó la metodología propuesta por la UCI para la actividad productiva, la cual adapta el ciclo de vida definido. Consiste en una variación de la metodología AUP, se encuentra estructurada en tres fases que tributan a la definición de los procesos que exige CMMI-Dev. nivel 2 (Sánchez 2014).

Tabla 2: Ciclo de vida de la Metodología Variación AUP- UCI. Fuente: (Sánchez 2014).

Fases AUP	**Fases Variación AUP -UCI**	**Objetivos de las fases (Variación AUP -UCI)**
Inicio	Inicio	Durante el inicio del proyecto se desarrollan las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del
Construcción		

Transición		proyecto considerando los requisitos y la arquitectura. Se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
	Cierre	Se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.4.2 Ingeniería de requisitos

Desde la perspectiva del proceso del software, la ingeniería de requisitos es una de las acciones importantes de la ingeniería de software que comienza durante la actividad de comunicación y continúa en la de modelado. Debe adaptarse a las necesidades del proceso, del proyecto, del producto y de las personas que hacen el trabajo (Pressman 2010).

La ingeniería de requisitos se entiende además, como todas las actividades relacionadas con la identificación y documentación de las necesidades de clientes y usuarios; creación de un documento que describe la conducta externa y las restricciones asociadas de un sistema que satisface dichas necesidades; análisis y validación del documento de requisitos para asegurar consistencia, compleción y viabilidad; evolución de las necesidades (Bravo et al. 2008; Brown 1999).

Los requisitos del sistema especifican qué es lo que el sistema debe hacer (funciones) y sus propiedades esenciales y deseables, esta actividad requiere consultas con los usuarios finales y clientes (Sommerville 2005).

Para el desarrollo de la solución siguiendo el escenario *Historias de Usuario* de la Variación AUP-UCI, se consideró la ejecución de los procesos: obtención y análisis, especificación, validación y administración de requisitos.

Una vez identificada la metodología a emplear, su escenario y los elementos básicos para la gestión de los requisitos, constituyó esencial definir las pautas arquitectónicas para desarrollar la solución.

1.4.3 Patrón arquitectónico

Los patrones arquitectónicos son la respuesta a los problemas de arquitectura en los productos de software. Realizan una descripción de los elementos y de la relación existente entre ellos, junto a un grupo de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones (Buschmann et al. 2007).

En el contexto de la investigación, atendiendo a que se trabajó sobre una versión previa ya desarrollada bajo el paradigma del patrón arquitectónico Modelo-Vista-Controlador (MVC) y el marco de trabajo Symfony2, se consideró consecuente continuar el desarrollo de la versión 2.0 del módulo de GOE del SAEF3 asumiendo las decisiones tecnológicas del equipo de desarrollo de la versión inicial.

Patrón arquitectónico Modelo-Vista-Controlador (MVC)

El patrón MVC surge con el objetivo de reducir el esfuerzo de programación, a partir de estandarizar el diseño de las aplicaciones. Es un paradigma que divide las partes que conforman la aplicación en el Modelo, las Vistas y los Controladores, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de *frameworks* basados en el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores (Díaz and Fernández 2012).

- **Vista:** presenta el modelo (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho modelo la información que debe representar como salida. Es la página HTML.
- **Controlador:** es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas.
- **Modelo:** es un conjunto de clases que representan la información del mundo real o que el sistema debe procesar sin tomar en cuenta ni la forma en la que esa información va a ser mostrada ni los mecanismos que hacen que esos datos están dentro del modelo, es decir, sin tener relación con ninguna otra entidad dentro de la aplicación.

1.4.4 Marco de trabajo Symfony2

Symfony2 basa su funcionamiento interno en el patrón arquitectónico MVC. En la figura 1 se describe el funcionamiento de este patrón arquitectónico en el marco de trabajo utilizado.

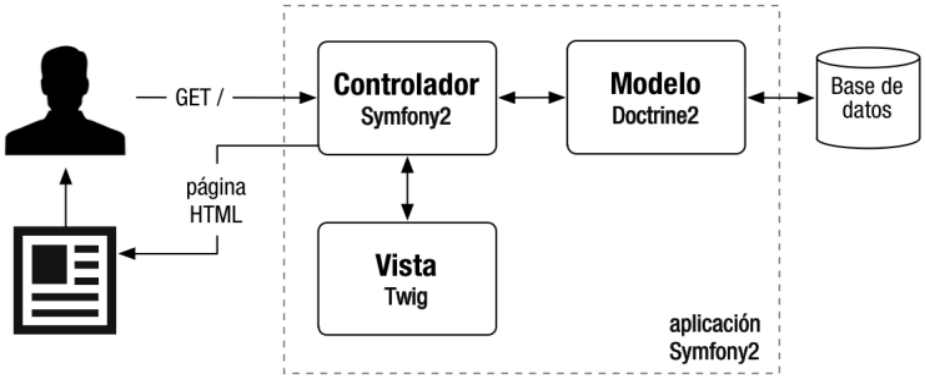

Figura 1: Funcionamiento general del patrón MVC. Fuente: (Potencier 2016).

Symfony es un completo marco de trabajo diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja como el ORM Doctrine, el componente formulario que soporta la validación automática de los datos y el manejo de caché reduce la carga del servidor y disminuye el tiempo de respuesta de las peticiones del usuario. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación (Potencier 2016) .

El sistema de enrutamiento determina qué *Controlador* está asociado con la página de la portada. Symfony2 ejecuta el *Controlador* asociado a esa portada y a su vez el *Controlador* solicita al *Modelo* los datos necesarios. El *Modelo* devuelve los datos, el *Controlador* solicita a la *Vista* la construcción de una página mediante una plantilla y que la inserción de los datos del *Modelo*. Por último el *Controlador* entrega al servidor la página creada por la *Vista* (Potencier 2016).

Determinados el patrón arquitectónico y el marco de trabajo para la solución, fue necesario realizar un estudio de los patrones de diseño para garantizar la elaboración de un modelo adecuado para su posterior implementación.

1.4.5 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, y describe la esencia de la solución a ese problema, de tal modo que pueda utilizarse esta solución un millón de veces más, sin siquiera hacerlo de la misma manera dos veces (Larman 1999; Pressman 2010).

Patrones Generales de Software para la Asignación de Responsabilidades (GRASP por sus siglas en inglés)

Los patrones de diseño GRASP, describen los principios fundamentales de la asignación de responsabilidades a objetos. Entre los más conocidos y empleados se encuentran (Larman 1999; Pressman 2010):

- Experto: cada clase tiene la responsabilidad de utilizar la información en la cual es experta y domina para realizar la labor para la que fue concebida. Tal es el caso de las clases del modelo que son las encargadas de toda la lógica del acceso a los datos.
- Creador: identifica quién debe ser el responsable de la creación de nuevos objetos o clases, donde la nueva clase deberá ser creada por la clase que tiene toda la información necesaria para realizar la acción, que usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de clase y contiene o agrega la clase.
- Alta cohesión: cada clase tiene como responsabilidad fundamental realizar las labores que solo le competen y que no son desempeñadas por otros elementos del diseño. Todas las clases están agrupadas por las funcionalidades que realizan.
- Bajo acoplamiento: las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista, lo que proporciona que la dependencia en este caso sea baja.
- Controlador: las clases controladoras se encargan de atender todas las peticiones y pasar los datos de la misma a las clases del modelo para su procesamiento. Al mismo tiempo son las clases que se encarga de enviar las respuestas a la vista.

Patrones Banda de los 4 (GoF por sus siglas en inglés)

Adaptados para solucionar un problema común en particular mediante la descripción de las clases y la

comunicación entre objetos; se agrupan en tres categorías: creacionales, estructurales y de comportamiento (Gamma et al. 2002; Larman 1999).

Creacionales:

- Inicialización vaga: refleja la delegación de la creación de un objeto o el cálculo de un valor hasta que sea realmente necesitado.

Estructurales:

- Adaptador: permite que la interfaz de una clase existente pueda ser usada desde otra interfaz sin necesidad de cambiar su código. Como ejemplo de ello está el motor de plantillas Twig donde cada plantilla redefine los bloques a su beneficio sin necesidad de cambiar la base.

Comportamiento:

- Iterador: provee una manera de acceder a los elementos de un objeto sin necesidad de exponer su representación interna, facilitando las iteraciones y el filtrado sobre colecciones de objetos.

Una vez diseñada la solución es necesaria la implementación de la misma, para ello es importante considerar estándares de codificación que faciliten la comprensión del código y su limpieza, para un correcto mantenimiento que se realice posteriormente a la solución.

1.4.6 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y, posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener (Microsoft 2016).

Symfony2 sigue los estándares de codificación del lenguaje PHP definidos internacionalmente (Potencier 2016).

1.4.7 Calidad de Software

Según IEEE (IEEE 1990) calidad del software se define como el grado con que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario. Humphrey (Humphrey 1995) coincide con IEEE respecto a que el principal objetivo de cualquier definición de calidad de software debe reflejar “las necesidades del cliente”. Por su parte Pressman (Pressman 2010) concuerda con las definiciones anteriores respecto a la satisfacción del cliente al definirla como un proceso eficaz de software cuya aplicación genera un producto útil considerando siempre la perspectiva del usuario. Además, Pressman incorpora un elemento sustancial referente a la medición del producto, pues establece que la calidad del software proporciona un valor medible para los que lo producen y lo usan que se pueden distinguir entre requerimientos y necesidades.

La calidad en el proceso de desarrollo de software debe medirse en cada una de sus actividades para poder detectar desde etapas tempranas posibles errores que constituyan muy costosos de implementar en etapas posteriores.

Métricas para la validación de requisitos

Existen diversas métricas para la validación de los requisitos entre ellas se encuentran (Garcia et al. 2013)

- Medición del atributo de calidad No ambiguo:

La medición de realiza a partir del porcentaje de requisitos interpretados de una única manera:

Q_1=nui/nr

Donde:

nui = número de requisitos con una única interpretación,

nr = número total de requisitos documentados.

Cuando el valor resultante se aproxima a 0, se interpreta como que cada requisito tiene múltiples interpretaciones; cuando el valor se aproxima a 1, se interpreta que cada requisito tiene una única interpretación. Se recomienda que $Q_1 = 1$.

- Medición del atributo de calidad *Comprensible*:

Una especificación de requisitos es comprensible cuando todos los que la leen (clientes, usuarios, desarrolladores) comprenden el significado de todos los requisitos con un mínimo de explicación.

Q_2 = nur/nr

Donde:

nur = número de requisitos que todos los revisores entienden

nr = número total de requisitos documentados

El valor resultante oscila en el rango entre 0 (ningún requisito comprendido) y 1 (todos los requisitos comprendidos). Se recomienda que Q_3 = 1.

- Medición del atributo de calidad *Correcto*:

Este atributo puede ser medido de la siguiente manera:

Q_3=nc/nr

Donde:

nc= número de requisitos correctos

nr= número total de requisitos documentados

Métricas para la validación del diseño

Una métrica es una medida efectuada sobre los programas, documentación, su desarrollo y mantenimiento que permite medir de forma cuantitativa la calidad de los atributos internos del software (Soto and Pompa 2015).

Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño de las clases se centran en cálculos de atributos y de operaciones para una clase individual, las métricas orientadas a la herencia analizan la forma en que las operaciones se reutilizan en la jerarquía de clases, las métricas para valores internos inspeccionan la cohesión, los aspectos orientados al código y las métricas orientadas a valores externos verifican el acoplamiento y reutilización entre las clases. Una de las métricas más empleadas es Tamaño de Clases (TC), pues permite medir el total de atributos y operaciones encapsulados en una clase para valorar la sobrecarga de responsabilidades asignadas. Valores grandes de TC indican que una clase puede tener demasiada responsabilidad, lo cual afecta la reutilización entre las clases y complica la implementación (Lorenz and Kidd 1994; Pressman 2010).

Chidamber y Kemerer proponen otro conjunto de métricas entre las cuales se encuentran: Métodos Ponderados por Clases (MPC), Árbol de Profundidad de Herencia (APH), Numero de Descendientes (ND), Acoplamiento entre Clases (CBO), Respuesta para una clase (RPC) y Carencia de Cohesión (LCOM) (Chidamber and Kemerer 1994; Pressman 2010).

Para la investigación se considera oportuno aplicar de Lorenz y Kidd la métrica TC, para evaluar si se distribuyen correctamente las asignaciones de responsabilidades entre las clases, verificándose así la cohesión entre las mismas, y de Chidamber y Kemerer las métricas CBO para verificar el grado de acoplamiento entre clases y LCOM para comprobar la cohesión entre las clases.

La siguiente tabla se muestra el resumen de las métricas a aplicar para validar el diseño de la solución.

Tabla 3: Métricas del diseño. Fuente: (Elaboración propia).

Métrica	Aplicación
TC	TC = T/CC T = total de operaciones de cada una de las clases CC = cantidad de clases
CBO	CBO = número de clases acopladas a otra clase mediante el uso de métodos a través de los objetos instanciados en ella. No es deseable que COB>14 ya que cuanto más alto es el resultado de dicha métrica, el código se considera propenso a fallos, de difícil mantenimiento, baja modularidad, no reutilizable y con un bajo nivel de encapsulamiento.
LCOM	LCOM = 0 cuando no existen métodos que tienen a un mismo atributo en común. LCOM = X X: cantidad de métodos que poseen uno o varios atributos en común. Es necesario obtener bajos valores de LCOM para garantizar una alta cohesión. Valores altos de LCOM indican niveles de complejidad elevados, señalando que las responsabilidades de la clase deben dividirse. Indican además, que el diseño es propenso a errores en el desarrollo y que la clase presenta bajos niveles de encapsulamiento.

Estrategia de pruebas de software

La estrategia de pruebas de software describe el enfoque y los objetivos generales de las actividades de prueba. Incluye los niveles de prueba, el tipo de prueba a ser ejecutada, los métodos de prueba y los casos de prueba diseñados para lograr los objetivos (Pressman 2010).

Niveles de prueba:

- Pruebas de desarrollador: son diseñadas e implementadas por el equipo de desarrollo.
- Pruebas independientes: son diseñadas e implementadas por una persona independiente al grupo de desarrolladores. El objetivo es proporcionar una perspectiva diferente y en un ambiente más rico que el de los desarrolladores.
- Pruebas de unidad: están enfocadas a los elementos verificables más pequeños del software. Son aplicables a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que funcionan como se espera. La prueba de unidad siempre está orientada a caja blanca.
- Pruebas de integración: son ejecutadas para asegurar que los componentes en el modelo de implementación operan correctamente cuando son combinados para ejecutar una funcionalidad. Se prueba un paquete o un conjunto de paquetes del modelo de implementación.
- Pruebas de sistema: se hacen para verificar el funcionamiento del software como un todo. Están dirigidas a verificar el programa final, después que todos los componentes de software y hardware han sido integrados.
- Pruebas de aceptación: es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por los usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

La metodología AUP propone específicamente los niveles:

- Pruebas internas: responden a los niveles inferiores a pruebas de aceptación.
- Pruebas de aceptación.

Para corroborar la efectividad de la solución se consideró oportuno aplicar los niveles de pruebas internas, específicamente: de desarrollador, independientes y de sistema; además, el nivel de prueba aceptación para corroborar la aceptación por el cliente.

Tipos de prueba:

Entre los tipos de pruebas que pueden aplicarse se encuentran (Hetzel 1993):

- Funcionalidad: verifican la habilidad del software para realizar el trabajo deseado.
- Fiabilidad: verifican la habilidad del software para mantenerse operativo.
- Eficiencia: verifican la habilidad del software para responder a una petición de usuario con la velocidad apropiada.
- Mantenibilidad: verifican la habilidad del software para realizar cambios en él rápidamente y con una adecuada proporción cambio/costo.
- Usabilidad: verifican la habilidad del software para satisfacer al usuario.
- Portabilidad: verifican la habilidad del software para correr en diferentes entornos informáticos.

Para el contexto de la investigación se consideró oportuno aplicar pruebas de funcionalidad, usabilidad y portabilidad.

Métodos de prueba:

Pruebas de caja negra: verifican el funcionamiento del software a través del cumplimiento de los requisitos funcionales. Permiten identificar:

- Funciones incorrectas o ausentes
- Errores de interfaz
- Errores en estructuras de datos o en accesos a las bases de datos externas
- Errores de rendimiento
- Errores de inicialización y terminación

Pruebas de caja blanca: evalúan la ejecución de cada sentencia del programa. Deben garantizar como mínimo que:

- Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas.
- Se ejerciten las estructuras internas de datos para asegurar su validez.

Para verificar el funcionamiento del módulo GOE versión 2.0 se consideró la aplicación de pruebas de caja negra para corroborar a partir de los casos de pruebas diseñados el funcionamiento en tiempo de

ejecución de la solución implementada.

Técnica Iadov:

La técnica Iadov mide la satisfacción del cliente con un producto, se compone de al menos cinco preguntas claves: tres cerradas y dos abiertas. Se establecen las preguntas y se conforma el cuadro lógico de Iadov, el resultado de la interrelación de las preguntas, indica la posición en la escala de satisfacción (Kuzmina 1970).

La escala de satisfacción está dada por los criterios:

1. Máxima satisfacción.
2. Más satisfecho que insatisfecho.
3. No definida.
4. Más insatisfecho que satisfecho.
5. Máxima insatisfacción.
6. Contradictoria.

Pregunta cerrada 3	Pregunta cerrada 1								
	No			No sé			Sí		
	Pregunta cerrada 2								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

Tabla 4: Cuadro lógico de Iadov. Fuente: (Kuzmina 1970).

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y - 1 de la siguiente forma:

Índice de satisfacción	Escala
Máxima satisfacción	+1
Más satisfecho que insatisfecho	0,5
No definido y contradictorio	0
Más insatisfecho que satisfecho	-0,5
Máxima insatisfacción	-1

Tabla 5: Índice de satisfacción de Iadov. Fuente (Kuzmina 1970).

La satisfacción grupal (ISG) se calcula por la siguiente fórmula:

$$\text{ISG} = \frac{A(+1) + B(+0{,}5) + C(0) + D(-0{,}5) + E(-1)}{N}$$

Donde:

- ✓ A representa el número de sujetos con índice individual 1
- ✓ B representa el número de sujetos con índice individual 2
- ✓ C representa el número de sujetos con índice individual 3 o 6
- ✓ D representa el número de sujetos con índice individual 4
- ✓ E representa el número de sujetos con índice individual 5
- ✓ N representa el número total de sujetos del grupo

1.5 Herramientas y tecnologías a utilizar

Para el desarrollo de la versión 2.0 del módulo GOE del sistema SAEF3 se continúa con el uso de las mismas herramientas y tecnologías empleadas en el desarrollo de su primera versión, las cuales son: Symphony2 como marco de trabajo, basado en el lenguaje de programación PHP en su quinta versión, JavaScript, UML 2.0 como lenguaje de modelado, como gestor de bases de datos PostgresSql y la aplicación pgAdmin3 para administrar dicho gestor, como servidor web Apache 2, Boostrap en el estilo CSS y PHP Storm como entorno de desarrollo integrado (IDE por sus siglas en inglés).

1.5.1 Servidor web: Apache server

Apache es un servidor web de Protocolo de Transferencia de Hipertextos (HTTP por sus siglas en inglés) de código abierto, para plataformas Unix (BSD, GNU/Linux, entre otros), Microsoft Windows, Macintosh entre otras. Es un proyecto de la Fundación de Software Apache, con el objetivo de suministrar un servidor seguro, eficiente, y extensible que proporcione servicios HTTP en sincronía con los estándares HTTP actuales. Presenta entre sus características que es: altamente configurable, posee bases de datos de autenticación, es modular, de código abierto y multi-plataforma (Apache 2011).

1.5.2 Lenguajes de programación

Debido a que la aplicación está orientada a la web y la arquitectura utilizada en la misma es la arquitectura Cliente/Servidor, se utilizara para la programación del lado del cliente el lenguaje interpretado HTML en su versión 5 y para la programación del lado del servidor PHP ya que este es un lenguaje que el propio servidor interpreta y ejecuta.

HTML5

HTML siglas de Hypertext Markup Language es un lenguaje de marcado para la elaboración de documentos web (web pages), define una estructura básica y un código. Cada etiqueta HTML describe diferentes contenidos dentro del documento web, como texto, imágenes, vídeos, entre otros. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación (Refsnes-Data 2016).

Boostrap (CSS)

Para la confección del visual se utilizó Boostrap en su versión 3 ya que es un potente framework front-end para el desarrollo de los estilos web, básicamente estructura por clases todo el estilo que se necesite aplicar al contenido web ya desarrollado, siendo la mejor forma de separar el contenido de la página, de su presentación, lo cual mejora la accesibilidad desde disimiles dispositivos y facilita el mantenimiento y perfeccionamiento del contenido (Refsnes-Data 2016).

JQuery 1.9

jQuery es una librería de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web (JQuery-Fundation 2016; Refsnes-Data 2016).

AJAX

AJAX, acrónimo de JavaScript asíncrono y XML por sus siglas en inglés (Asynchronous JavaScript And XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones (JQuery-Fundation 2016; Refsnes-Data 2016).

PHP 5.4.4-14

Es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Es muy rápido, pues permite al equipo de desarrollo la generación dinámica de páginas, su integración con las bases de datos y el servidor Apache. PHP es multiplataforma y funciona tanto para Unix como para Microsoft Windows, de manera que el código creado para Unix no sufre modificaciones al ser ejecutado en Windows y viceversa. Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones web gracias a la extensa librería de funciones con la que está dotado (Refsnes-Data 2016).

1.5.3 Entorno de Desarrollo Integrado (IDE)

PhpStorm 9.0

Incorpora tecnologías emergentes para disfrutar el desarrollo web con un entendimiento profundo del código y soporte avanzado para ambientes remotos. PhpStorm ofrece un editor enriquecido e inteligente para PHP que comprende el código y su estructura, soportando PHP 5.3, 5.4, 5.5 & 5.6 para proyectos modernos y legados. El IDE provee finalización inteligente de código, señalamiento de sintaxis, configuración extendida de formato de código, chequeo de errores al instante, plegado de código, soporte para mezclas de lenguajes entre otros (JetBrains 2016).

1.5.4 Sistema Gestor de Bases de Datos (SGBD)

Un SGBD es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones, permite definir datos a distintos niveles de abstracción, la manipulación de dichos datos, así como garantizar la integridad y seguridad de dichos datos.

PgAdmin3

Es la herramienta de código abierto de administración para la base de datos PostgreSQL. Posee como características: edición rápida de consultas, soporte para todos los tipos de objetos de PostgreSQL. Está diseñado para dar respuesta a las necesidades de la mayoría de los usuarios, desde la escritura de consultas simples en SQL hasta el desarrollo de bases de datos complejas. La interfaz gráfica soporta las características presentes de PostgreSQL. Está disponible en varios lenguajes y para varios sistemas operativos (pgAdmin 2016; PostgreSQL 1996).

PostgreSQL v9.4

Es un servidor de base de datos relacional libre. Permite al equipo de desarrollo la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Cuenta con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y es multiplataforma (PostgreSQL 1996).

1.5.5 Marco de Trabajo (Framework)

Symfony 2.3

El marco de trabajo o framework permite simplificar el desarrollo de un sistema mediante la utilización de distintos tipos de lenguajes, el uso de componentes, librerías y estructuras de acuerdo a los diferentes patrones existentes. Aporta agilidad y facilidad a los programadores, pues encapsula operaciones complejas en instrucciones sencillas y posee una organización bien estructurada del proyecto informático en desarrollo. Symfony2 es un marco estable para desarrollar aplicaciones, está publicado bajo una licencia de software libre y es fácil de instalar y configurar en la mayoría de las plataformas. Es sencillo de usar y al mismo tiempo es flexible. Su arquitectura interna está completamente desacoplada, lo que permite adaptarse fácilmente a los diferentes proyectos. Posee la ventaja de ser multiplataforma y compatible con la mayoría de gestores de bases de datos como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Hace uso del patrón arquitectónico MVC y sigue la mayoría de las mejores prácticas y patrones de diseño para la web (Potencier 2016).

1.5.6 Herramientas CASE (Computer Aided Software Engineering)

Visual Paradigm 8.0

Es una herramienta que facilita y permite al equipo de desarrollo visualizar, diseñar, integrar y distribuir la aplicación que se necesita. Permite una rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. La herramienta CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos UML (Visual-Paradigm 2016).

Lenguaje Unificado de Modelado (UML)

Se empleó UML en su versión 2.0 como lenguaje de modelado, ya que el mismo permite al equipo de desarrollo visualizar, especificar, construir y documentar los artefactos del sistema, además de estandarizar toda la información generada sobre las funcionalidades del módulo, las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes (Rumbaugh et al. 2000).

1.6 Conclusiones parciales

En el presente capítulo, se realizó un análisis de conceptos asociados al campo de acción a fin de facilitar la comprensión en el proceso de desarrollo de la solución. El estudio realizado de los sistemas homólogos, identificó que ninguna de las propuestas satisface las necesidades del VDEA de la Facultad 3, por lo que se requirió la modificación de funcionalidades ya existentes y la incorporación de nuevas funcionalidades al módulo GOE del SAEF3.

Para el desarrollo de la solución se constató la necesidad de emplear la metodología AUP en su variación UCI y, además, la aplicación de métricas para evaluar los requisitos y el diseño desarrollado. Se consideró oportuno como parte de la estrategia de pruebas aplicar los niveles de pruebas internas y de aceptación, los tipos de pruebas funcionalidad, usabilidad y portabilidad, y el método de prueba de caja negra. Para medir la satisfacción del cliente con la solución se consideró aplicar la técnica Iadov.

Para desarrollar la solución, se ratificó el empleo de tecnologías como: servidor web apache server, HTML5, PHP5.4.4-14, PhpStorm 8, PostgreSQL v9.4, pgAdmin3 v1.14.2-2, Symfony 2.3.8, Bootstrap 3, jQuery1.9, Visual Paradigm 8.0 y UML, las cuales fueron empleadas por el equipo de desarrollo de la primera versión del módulo.

CAPÍTULO 2: MÓDULO GOE DEL SAEF3 V 2.0

2.1 Introducción

En el presente capítulo se definen los requisitos funcionales y no funcionales para el desarrollo del módulo y se aplican las métricas de validación de requisitos para corroborar el cumplimiento de los atributos de calidad. Se muestran además los prototipos de interfaz de usuario, los artefactos del diseño y los resultados de la aplicación de las métricas del diseño.

2.2 Descripción de la solución

Como solución a la problemática planteada, se propone la implementación de nuevas funcionalidades en el módulo GOE del sistema SAEF3 en su versión 2.0, así como la modificación de algunas de las funcionalidades ya existentes a fin de elevar el control de la información asociada a los procesos de planificación y control de la GOE en la Facultad 3. Para ello se identificaron los requisitos funcionales que posteriormente se describieron como historias de usuario. Se identificaron además, los requisitos no funcionales de la solución.

2.2.1 Requisitos funcionales

Requisitos funcionales a modificar:

RF_1: Obtener Histórico de las guardias planificadas.

RF_2: Configurar rotaciones de los turnos en la planificación de la guardia.

RF_3: Asignar roles de usuarios en el sistema.

RF_4: Planificar guardia.

RF_5: Mostrar factibilidad de guardia configurada.

RF_6: Gestionar restricción por grupo.

RF_7: Gestionar restricción del personal

RF_7.1: Modificar restricción del personal

RF_7.2: Listar restricciones del personal

Nuevas funcionalidades a implementar:

RF_8: Autenticar usuario mediante consumo de servicios.

RF_9: Gestionar potencial de la guardia.

RF_9.1: Adicionar potencial de la guardia mediante consumo de servicios.

RF_9.2: Listar potencial de la guardia.

RF_9.4: Eliminar potencial de la guardia.

RF_10: Gestionar mensajes de notificación del sistema.

RF_10.1: Adicionar mensaje de notificación del sistema.

RF_10.2: Listar mensajes de notificación del sistema.

RF_10.3: Editar mensaje de notificación del sistema.

RF_10.4: Eliminar mensaje de notificación del sistema.

RF_11: Notificar la planificación mediante correo a usuarios registrados en el sistema.

RF_12: Exportar a Excel la planificación de la guardia.

RF_13: Exportar a Excel el registro de cumplimiento.

RF_14: Gestionar solicitudes de cambio de guardia.

RF_14.1: Adicionar solicitud de cambio de guardia.

RF_14.2: Listar solicitudes de cambio de guardias realizadas.

RF_14.3: Cancelar solicitud de cambio de guardia.

RF_15: Intercambiar planificación de guardia entre personas.

RF_16: Mostrar detalles de la planificación de la guardia.

RF_17: Gestionar cumplimiento de la guardia:

RF_17.1: Adicionar cumplimiento de la guardia.

RF_17.2: Listar cumplimiento de la guardia.

RF_17.3: Editar cumplimiento de la guardia.

RF_18: Notificar incumplimiento de guardia a personal implicado.

RF_19: Notificar registro de cumplimiento de la guardia a jefes de área.

2.2.2 Requisitos no funcionales

Funcionalidad:

Seguridad:

- El sistema SAEF3 versión 2.0 maneja la seguridad de acceso y administración de usuarios mediante la asignación de roles.
- Los usuarios deben autenticarse para acceder a las funcionalidades del módulo.
- El acceso a las funciones será restringido, atendiendo a los niveles asignados según el rol del usuario.

Instalabilidad:

- El sistema podrá ser instalado en el ambiente especificado en los requisitos de hardware para los servidores.

Hardware:

Servidor Aplicación:

- Procesador: 3.40 GHZ
- RAM: 4 GB

- Disco duro: 160 GB

Servidor de Base de Datos:

- Procesador: 3.00 GHZ
- RAM: 2 GB
- Disco duro: 160 GB

PC Cliente

- Procesador: 1.40 GHZ
- RAM: 1 GB

2.3 Historias de usuario

En la presente sección se muestran las especificaciones de algunas historias de usuario del módulo GOE versión 2.0 del SAEF3, el resto se anexa al documento.

Historia de Usuario	
Número: 4	**Nombre del requisito:** Planificar guardia
Programador: Osvaldo Alejandro Pacheco Morales	**Iteración Asignada:** 1
Prioridad: alta	**Tiempo estimado:** 10 horas
Riesgo en desarrollo: alto	**Tiempo real:** 11 horas
Descripción: Permite realizar la planificación del personal en un área determinada, para un rango de fechas establecido. Además, se especifica el tipo de planificación lo que impacta directamente en la selección del personal al que se le va a planificar la guardia.	
Observaciones: Si el personal a planificar no suple los turnos y postas configurados para el rango de fechas, no se completa la planificación y se sugiere revisar la factibilidad de configuración de la guardia.	
Prototipo elemental de interfaz gráfica de usuario:	

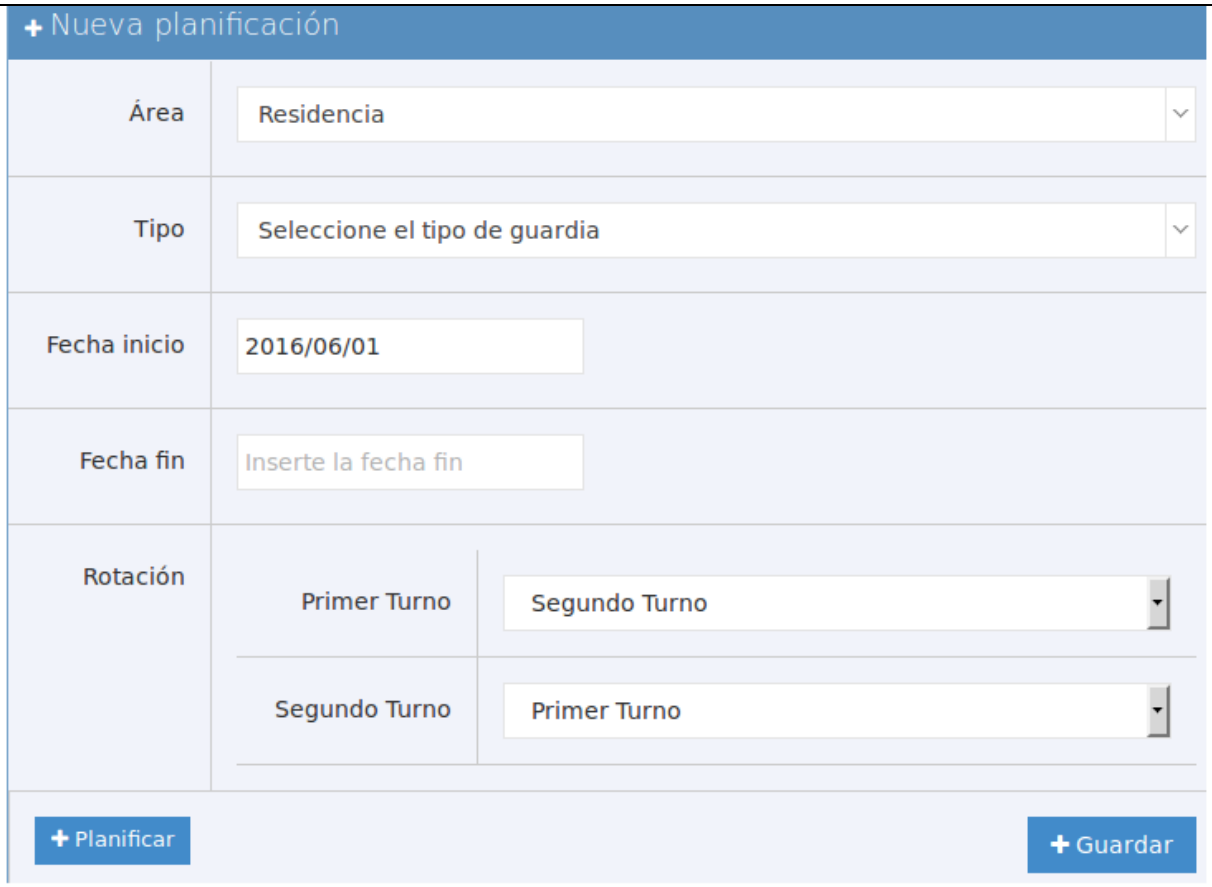

Figura 2: Prototipo de interfaz de usuario “Planificar guardia”. Fuente: (Elaboración propia).

2.4 Validación de requisitos

Para realizar la validación de los requisitos se tuvo en cuenta no solo el criterio de los desarrolladores, sino también el criterio del cliente y de sus homólogos. Los resultados obtenidos se muestran a continuación.

Atributos de calidad	Resultado	Interpretación
No ambiguo	$Q_1 = \frac{n_{ui}}{n_r} = \frac{29}{29} = 1$	Todos los requisitos tienen una única interpretación.

Comprensible	$Q_2 = \frac{n_{ur}}{n_r} = \frac{29}{29} = 1$	Todos los requisitos son comprendidos.
Correcto	$Q_3 = \frac{n_c}{n_r} = \frac{29}{29} = 1$	Todos los requisitos representan una necesidad de función del sistema a construir.

Tabla 6: Resultados de la aplicación de las métricas de validación de requisitos. Fuente: (Elaboración propia).

A partir de la aplicación de las métricas de validación de requisitos se pudo constatar que existe un 100% de concordancia respecto al cumplimiento de los atributos de calidad de los mismos, obteniéndose que en su totalidad son: no ambiguos, comprensibles y correctos.

2.5 Diseño de la solución

En el presente epígrafe se aplican a la solución los patrones del diseño especificados, obteniéndose el diagrama de clases del diseño con estereotipos web y el modelo de datos.

2.4.1 Diagrama de clases del diseño con estereotipos web

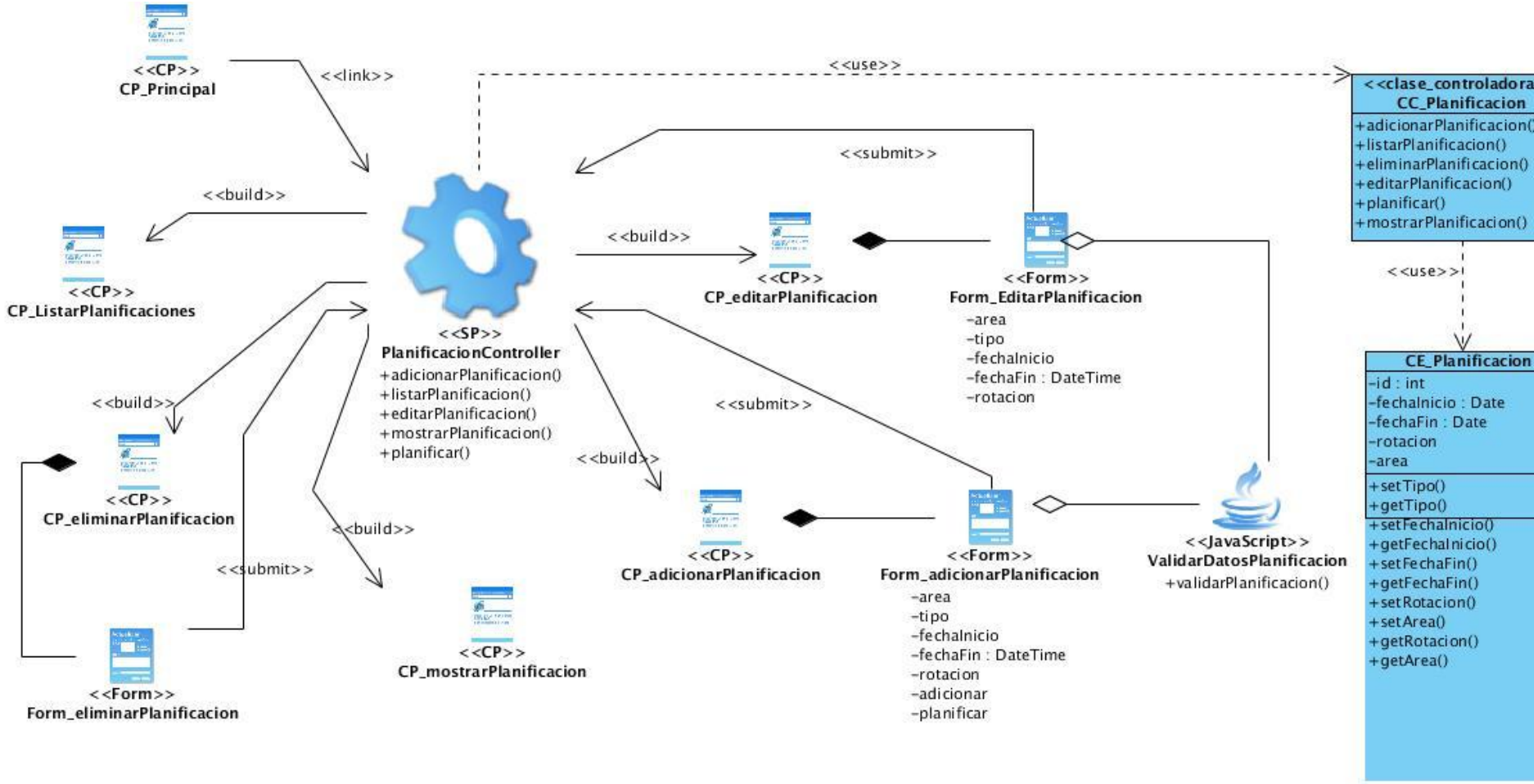

Figura 3: Diagrama de clases del diseño con estereotipos web del RF_4: Planificar guardia. Fuente: (Elaboración propia).

Patrones GRASP:

- Experto: indica que la responsabilidad de la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria. En el diagrama se evidencia que la clase *CE_Planificación* es experta en la información y se le asigna la responsabilidad de insertar y obtener la información asociada a sus atributos.
- Creador: se refleja en las clases controladoras donde se encuentran las acciones definidas para las operaciones lógicas del negocio referentes a la instanciación de entidades y la creación de las páginas clientes.
- Controlador: contiene toda la lógica que la aplicación necesita para generar el contenido de la página, y la respuesta puede ser una página HTML, un documento XML, un array JSON serializado, una cabecera de error. Permite que se procesen todas las peticiones de manipulación por analizar a través de un objeto de controlador único. Estas peticiones web son tratadas por un solo controlador frontal que posee el marco de trabajo llamado “app.php” siendo este el único punto de entrada a la aplicación.
- Alta cohesión y Bajo acoplamiento: en todas las clases se muestra una alta cohesión y un bajo acoplamiento ya que cada una de ellas posee la responsabilidad de realizar las labores que solo le competen, favoreciendo que la dependencia sea baja. La correcta distribución y separación de responsabilidades aplicando los patrones Experto, Creador y Controlador, contribuyen a que no existan clases sobrecargadas, a que funcionen cohesiva y armónicamente, y a disminuir el grado de dependencias sobre una misma clase.

Patrones GoF

- Estructurales: Adaptador

El patrón Adaptador se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda. Symfony2 ofrece la estructura para aplicar el patrón adaptador a través de la herencia de plantillas, establece una base de la cual van a extender las demás plantillas y determina dentro de su estructura bloques que pueden ser sobrescritos en cada vista modificándose según el objetivo buscado por estas.

Creacionales:

- Inicialización vaga: refleja la delegación de la creación de un objeto o el cálculo de un valor hasta que sea realmente necesitado.

```
$p ='';
$em = $this->getDoctrine()->getManager();
$p = $em->getRepository('Fac3AppBundle:Personas')->find($persl->getId());
$notif = $em->getRepository('Fac3AppBundle:Notificaciones')->findBy(array(
```

Figura 4: Patrón inicialización vaga. Fuente: (Elaboración propia).

Comportamiento:

- Iterador: se utilizó para acceder a los elementos de un objeto para su representación en las vistas y funcionalidades; en las clases controladoras, necesarias para acceder a la información contenida en cada instancia del objeto dentro de la colección del tipo de objeto referido.

```
for ($n = 0; $n < count($p->getNotificaciones()); $n++) {
                    ld($jefesBD[0]->getNotificaciones()[$n]->getId());
                    ldd($noti[0]->getId());
    if ($p->getNotificaciones()[$n]->getId() == $notificacion[0]->getId()) {
        $flag = true;
        break;
```

Figura 5: Patrón iterador. Fuente: (Elaboración propia).

2.4.2 Modelo de datos

El diseño de la base de datos tiene como objetivo representar la información que se gestiona sin contemplar los mecanismos que existen para el acceder a esta o las vías para su almacenamiento. Se visualiza a través de un conjunto de entidades y las relaciones entre ellas, que permiten el almacenamiento de la información con un mínimo de redundancia, manteniendo su integridad y facilitando la recuperación de esta para su consulta. A continuación, se muestra el modelo de datos generado para la versión 2.0 del módulo GOE, el cual consta de 16 tablas persistentes, donde se representa (en color naranja) la información manejada por los procesos que se han mantenido sin modificación. Se observa (en color gris) la información generada por las funcionalidades modificadas para la versión 2.0 del módulo GOE; así como en color blanco la información generada por las nuevas funcionalidades implementadas para dicho módulo.

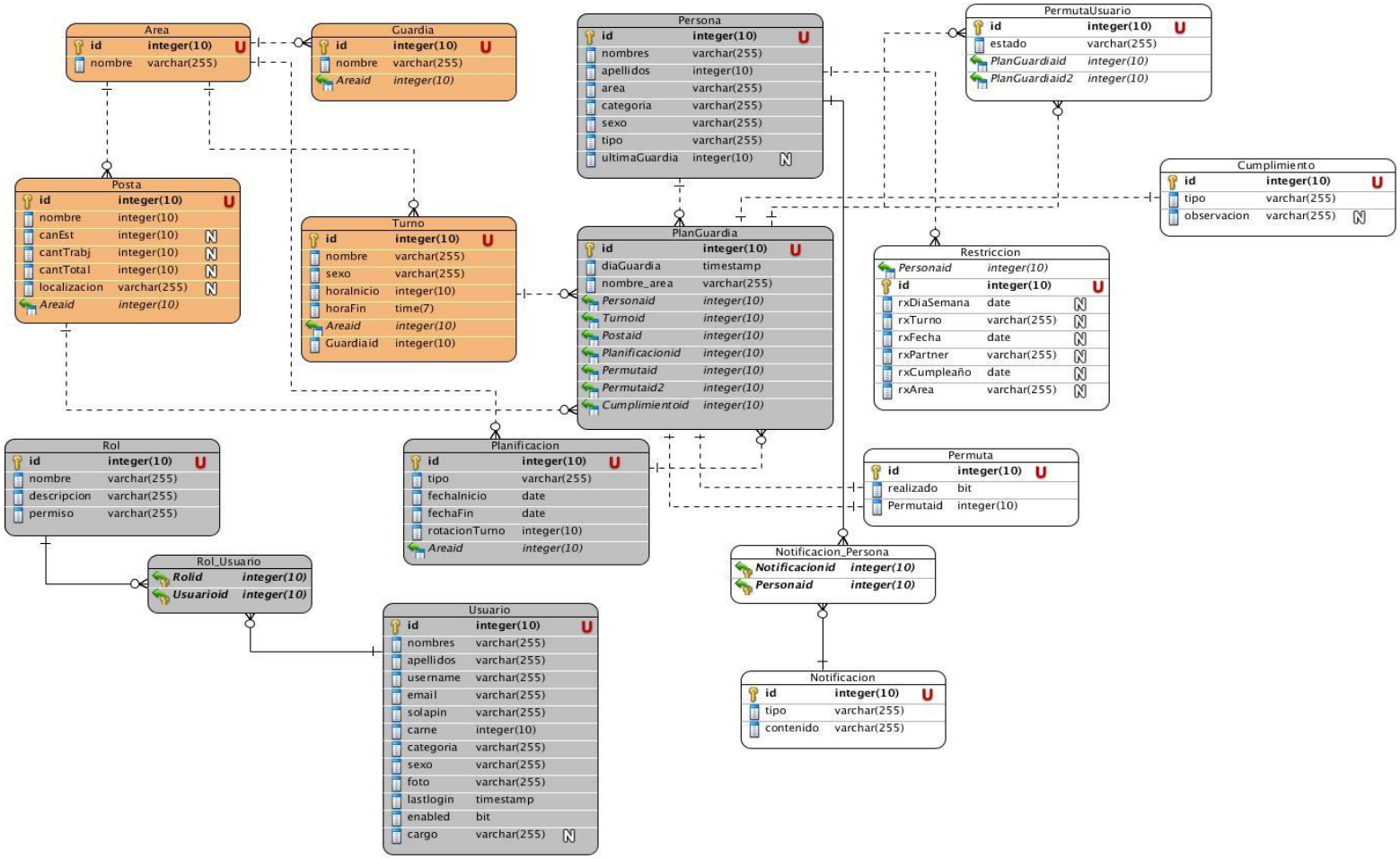

Figura 6: Modelo de datos. Fuente: (Elaboración propia).

2.4.3 Verificación del diseño

Para verificar la calidad del diseño de la solución se emplearon las métricas de TC, CBO y LCOM para medir los indicadores: responsabilidad, complejidad de implementación y reutilización.

Métrica	Resultados del análisis
CBO	La aplicación de la métrica arrojó resultados aceptables en cuanto a acoplamiento, ya que se observa en la clase DefaultController.php de un total de 23 métodos 12 son dependencias de otras clases, el resto no supera las 8 dependencias llegando a la conclusión que existen valores bajos de acoplamiento entre las clases.
TC	El análisis de la métrica dio como resultado que existe un total de 17 clases de las cuales 9 son no persistentes, dando un total de 218 métodos y arrojando un promedio de tamaño de clases de 12,8 aproximadamente 13 métodos por clases.
LCOM	El análisis arrojado tras aplicar dicha métrica dio como resultado que en la clase DefaultController.php que consta de 23 métodos de los cuales 12 comparten el mismo atributo del objeto persona, siendo este el resultado más bajo en cuanto a cohesión en una clase, el resto poseen un valores más elevados, ejemplo en la clase TurnoControlleren que de 10 métodos 8 comparten el mismo atributo, siendo este el comportamiento general en el resto de las clases, por lo que se concluye que existe altos valores de cohesión en las clases.

Tabla 7: Resultado de la aplicación de las métricas del diseño. Fuente: (Elaboración propia)

2.5 Conclusiones parciales

En el presente capítulo se describieron las características del sistema, para ello se identificaron requisitos funcionales y no funcionales y se realizó la especificación de los funcionales a través de historias de usuario. La aplicación de las métricas de validación de requisitos permitió corroborar que en su totalidad fueron no ambiguos, comprensibles y correctos.

Se realizó además el modelado del diseño, donde se pudo evidenciar la aplicación de los patrones del diseño. La aplicación de las métricas del diseño TC, RC y CCM confirmó la calidad del diseño realizado para facilitar la implementación del módulo lo cual se traduce en una baja responsabilidad y complejidad, que favorece la reutilización, la alta cohesión y el bajo acoplamiento.

CAPÍTULO 3 IMPLEMENTACIÓN Y ANÁLISIS DE LOS RESULTADOS

3.1 Introducción

En el capítulo se realizan las validaciones de las variables de la investigación para garantizar el cumplimiento del objetivo general planteado. Se exponen los resultados obtenidos mediante las pruebas realizadas al módulo GOE versión 2.0 del SAEF3, se describen los estándares de codificación, el tratamiento de errores aplicado y se muestra el diagrama de componentes que estructura la implementación.

3.2 Aspectos relevantes de la implementación

A continuación, se muestra el diagrama de componentes desarrollado, los estándares de codificación, empleados y el tratamiento de errores.

3.2.1 Diagrama de componentes

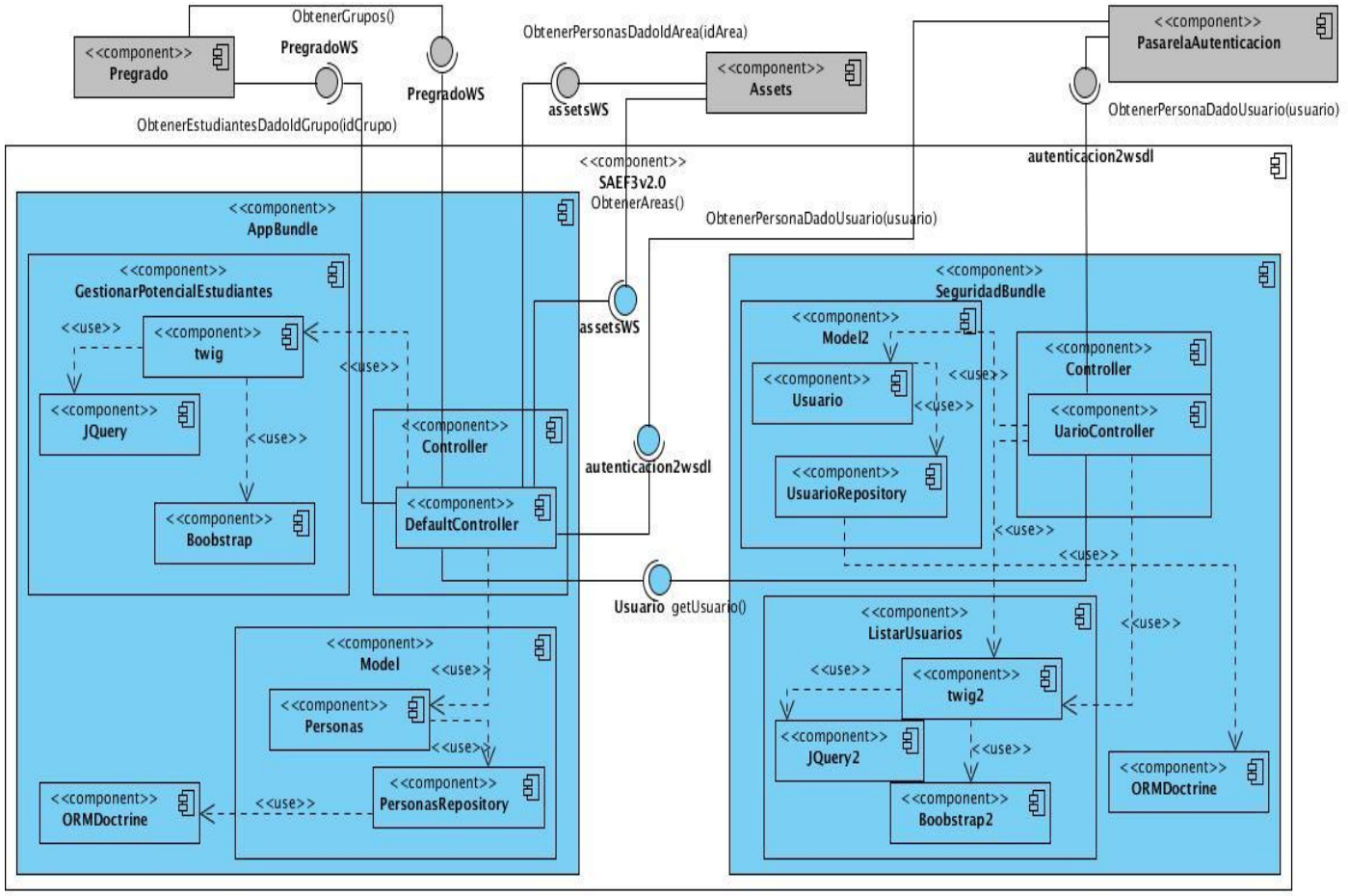

Figura 7: Diagrama de componentes módulo GOE versión 2.0 del SAEF 3.Fuente: (Elaboración propia).

El diagrama de componentes muestra los elementos de implementación que componen un sistema, refleja las interfaces provistas y requeridas, puertos y las relaciones entre estos. Un componente representa una parte de un sistema modular, desplegable y reemplazable, que encapsula la implementación y expone un conjunto de interfaces (Larman 1999; Larman 2003).

3.2.2 Diagrama de despliegue

El diagrama de despliegue UML se muestra la arquitectura física de un sistema informático, se representan los equipos, dispositivos, sus interconexiones y el software que se encontrará en cada máquina.

(Symfony 2015)Figura 8: Diagrama de componentes módulo GOE versión 2.0 del SAEF 3.Fuente: (Elaboración

propia).

En la figura 8 se muestra cómo será desplegado el SAEF3 al cual pertenece el módulo GOE. El diagrama de despliegue está compuesto por la conexión de la PC_Cliente al servidor de aplicaciones web Apache a través del protocolo de transferencia de hipertexto (HTTPS) por el puerto 8080 y con el servidor de Base de Datos PostgreSQL mediante el puerto 5432.

3.2.3 Estándares de codificación

Symfony 2.6, el framework empleado para el desarrollo del módulo GOE para el sistema SAEF3, se rige por los estándares de codificación definidos en los documentos PSR-0, PSR-1 y PSR-2, a continuación, se da muestra de dichos estándares (Symfony 2015).

Convención de Nombres:

- Utiliza camelCase y no guiones bajos, para variables, funciones y nombres de métodos.
- Utiliza guiones bajos para definir opciones, argumentos y nombres de parámetros.
- Utiliza los *namespace* para todas las clases.
- Utiliza caracteres alfanuméricos y guiones bajos para nombres de archivos.

Estructura

- Nunca utilices las etiquetas cortas (<?).
- No finalices las clases con la etiqueta usual de cierre (*?*>).
- La indentación se realiza utilizando cuatro espacios (tabulaciones no están permitidas).
- Utiliza el carácter de salto de línea (*0x0A*) para finalizar las líneas.
- Agrega un único espacio después de cada delimitador coma.
- No pongas espacios después de la apertura de un paréntesis y antes del cierre del mismo.
- Agrega un único espacio alrededor de operadores (==, *&&*,...).
- Agrega un único espacio antes de los paréntesis de apertura de una palabra clave de control (*if*, *else*, *for*, *while*,...).
- Agrega una línea en blanco antes de la sentencia *return.*
- No agregues espacios al final de las líneas.
- Utiliza llaves para indicar el cuerpo de las estructuras de control sin importar el número de sentencias que éstas contengan.

- Coloca las llaves en sus propias líneas para clases, métodos y declaración de funciones.
- Separa las sentencias condicionales y las llaves de apertura con un único espacio sin dejar una línea en blanco.
- Declara explícitamente la visibilidad de clases, métodos y propiedades (el uso de *var* está prohibido).
- Utiliza constantes de tipo PHP nativas en minúsculas: *false*, *true* y *null*. Lo mismo aplica para *array ().*
- Define una clase por archivo.
- Declara las propiedades de las clases antes de los métodos.
- Declara los métodos públicos primero, luego los protegidos y finalmente los privados.

3.2.4 Tratamiento de errores:

El framework viene con una página de error por defecto para la mayoría de los formatos de respuesta más comunes tales como: JSON (error.json.twig), XML (error.xml.twig), JavaScript (error.js.twig) y CSS (error.css.twig) y también otros menos comunes como TXT, RDF, y ATOM la cual es configurable y totalmente integrable a la estructura y el diseño (Symfony 2015).

Las validaciones realizadas en las vistas aseguran la entrada correcta de los datos a la aplicación. Su principal misión es evitar que datos incorrectos sean introducidos en los diferentes campos de los formularios y que datos inconsistentes o incorrectos lleguen al servidor. Para la notificación de errores en las interfaces del módulo se utilizan las validaciones que provee la librería jQuery. Por otro lado, el marco de trabajo Symfony contiene un conjunto de procedimientos que garantizan validaciones básicas de los campos en las entidades a disponibilidad del desarrollador denominadas asserts.

3.3 Validación de la solución

A continuación, se muestran los resultados obtenidos de la aplicación de las pruebas aplicadas al módulo.

3.3.1 Pruebas de caja negra

Las pruebas de caja negra se ejecutan sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa y evaluando su comportamiento mediante las entradas y salidas obtenidas a partir de los casos de prueba. Para la aplicación de esta prueba se seleccionó la técnica de Particiones de Equivalencia.

A continuación, se muestra un ejemplo de caso de prueba efectuado al requisito *Planificar guardia.*

Tabla 8: Caso de prueba "Planificar guardia". Fuente: (Elaboración propia).

Nombre del requisito	Descripción general	Escenario de Prueba	Flujos del sistema
Planificar	Se gestiona la planificación de la guardia	EP 1.1 Planificar	-Se selecciona el área a planificar -Se selecciona el tipo de planificación -Se inserta la fecha inicio y la fecha fin de la planificación -Se establece la configuración de la rotación de los turnos -Se presiona el botón planificar -Se presiona el botón guardar -Se regresa al listado de planificaciones.

Tabla 9: Descripción de variable. Fuente: (Elaboración propia).

No	Nombre de campo	Tipo	Válido	Inválido
N/A	N/A	N/A	N/A	N/A

Tabla 10: Juego de datos. Fuente: (Elaboración propia).

Id escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EP 1.1	Planificar	Se realiza la planificación de la guardia para el área, tipo, fechas y rotación seleccionados.	

3.3.2 Tipos de pruebas

Se aplicaron tres tipos de pruebas:

- Funcionalidad: esta prueba fue aplicada en todos los niveles de prueba mediante el uso del método de caja negra.
- Portabilidad: el módulo fue probado en diferentes plataformas (Nova 4, Windows 7 y 8.1), mostrando un desempeño aceptable.
- Usabilidad: el cliente y sus homólogos le aplicaron pruebas al módulo, a partir de estas pruebas que se realizaron mostraron su satisfacción con respecto al producto, la cual se refleja además, en los resultados de la aplicación de la técnica Iadov.

En una primera iteración se detectaron 22 no conformidades, de las cuales 4 no conformidades estuvieron asociadas al funcionamiento del sistema, 10 a problemas de ortografía y 8 a problemas con las notificaciones y mensajes de confirmación para las acciones de eliminación del sistema. En una segunda iteración se identificaron 9 no conformidades, de las cuales 6 estuvieron asociadas a faltas de ortografía y el resto vinculadas a notificaciones y mensajes de confirmación para las acciones de edición. En una tercera iteración no se detectaron no conformidades como se refleja en la siguiente figura.

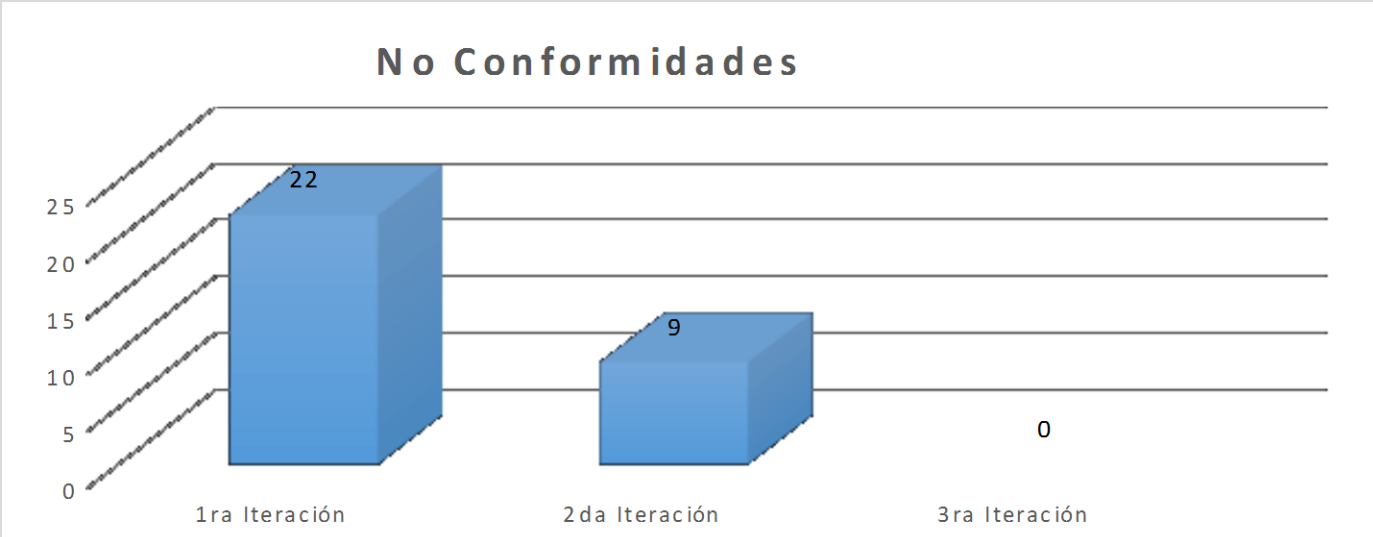

Figura 9: Resultado de aplicación de caso de prueba. Fuente: (Elaboración propia).

3.4 Técnica Iadov

Para valorar el grado de satisfacción del cliente con la solución desarrollada respecto al control de la información, se aplicó la técnica Iadov. Durante la valoración fungieron como clientes, los VDEA de todas las facultades de la UCI, así como 2 trabajadores del VDEA de la Facultad 3, para un total de 9 personas. Como resultado se obtuvo:

Tabla 11: Cuadro lógico de Iadov. Fuente: (Elaboración propia).

¿Considera factible utilizar la versión 2.0 del módulo GOE del SAEF3, para ejecutar los procesos asociados al área económica del VDEA?	¿Considera que el uso de la primera versión del módulo GOE satisface las necesidades de control de la información a pesar de las insuficiencias detectadas en él?								
	No			No sé			Sí		
	¿Considera que la versión 2.0 del módulo GOE refleja mejoras sobre el control de la información asociada al VDEA con respecto a la versión anterior?								
	Sí	No sé	No	Sí	No sé	No	Sí	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3

Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	6
No sé qué decir	2	3	6	3	3	3	6	3	4

$$ISG = \frac{8*1+1*0,5}{9}$$

De manera que el ISG = 0,94.

Los resultados de la satisfacción individual según las categorías empleadas fueron los siguientes:

Tabla 12: Resultado de aplicación de la técnica Iadov. Fuente: (Elaboración propia).

Nivel de satisfacción	Cantidad	%
Máxima satisfacción	8	88,89
Más satisfecho que insatisfecho	1	11,11
No definida	0	0,00

Al procesar las respuestas a las preguntas en el cuadro lógico de Iadov, se obtiene un grado de satisfacción grupal de 0,94; lo cual se traduce en una clara satisfacción con el uso del módulo GOE del SAEF3 en su versión 2.0.

En el criterio respecto al control y disponibilidad de la información en el VDEA a través del uso de la solución propuesta, hubo una concordancia de un 100% en que la primera versión del módulo no satisface las necesidades de control de la información. De igual manera el 100% de los participantes manifestó que le gustaría mucho hacer uso del módulo GOE del SAEF3 en su versión 2.0 para los procesos asociados a la planificación y control de la GOE. Mientras que un encuestado que representa el 11,11% no sabe si la versión 2.0 representa mejoras respecto a la versión anterior.

De las preguntas abiertas formuladas se obtuvieron las siguientes valoraciones positivas sobre el módulo del SAEF3 en su versión 2.0:

- La versión 2.0 del módulo GOE posibilita notificar el cumplimiento a implicados.
- Permite el intercambio de planificaciones entre usuarios del sistema.

- Realiza la planificación teniendo en cuenta las restricciones de los implicados, así como la rotación de los turnos y el rango de fecha seleccionado.
- Establece control de la guardia, así como brinda un sistema de notificación.

La aplicación de la técnica de Iadov aportó información significativa respecto al grado de satisfacción del cliente. Los resultados obtenidos y los criterios emitidos validan la fortaleza de la propuesta, reflejándose una valoración muy positiva del cliente con la solución.

3.4.1 Validación de la variable de la investigación

Para validar la variable de la investigación (tabla 12), se analizaron las dimensiones de las mismas. Estas dimensiones fueron seleccionadas a partir del concepto de control de la información.

Tabla 13: Validación de la variable de la investigación. Fuente: (Elaboración propia).

Variables	Dimensiones	Proceso actual	Proceso con la utilización del módulo implementado
Control de la información	Fiscalización	Actualmente los jefes de áreas no pueden visualizar la información resultante del registro del cumplimiento de la guardia de las personas involucradas en la planificación que están bajo su responsabilidad por lo que no puede tomar acciones en cuanto a los incumplimientos de sus subordinados.	Una vez realizado el registro del cumplimiento por el VDA., a los jefes de área les llega la notificación del registro del cumplimiento y podrán visualizar los cumplimientos del personal asociado a este exclusivamente.
	Comprobación	El registro del cumplimiento de guardia se notifica vía correo, mediante una relación que se hace llegar a cada jefe de área y se adjunta la relación de	Una vez registrado el cumplimiento el sistema automáticamente notifica a los jefes de área, quienes pueden acceder a dicha información y verificar el cumplimiento o incumplimiento de la

		subordinados y el respectivo cumplimiento de guardia.	guardia de sus subordinados.
		El análisis de la factibilidad se realiza sin tener en cuenta las restricciones del potencial de guardia, lo que provoca problemas en el momento de realizar la planificación de la guardia.	El sistema brinda la posibilidad de realizar un análisis previo de la configuración de las postas y turnos con respecto al potencial de guardia registrado, así como las restricciones que tienen registradas cada uno, evitando consigo problemas en la planificación realizada por el sistema, a causa de no contar con el potencial suficiente.

3.5 Conclusiones parciales

El tratamiento de errores minimizó la posibilidad del error humano y el uso de los estándares de codificación favoreció la limpieza del código y su comprensión para su posterior mantenimiento.

Con la aplicación de las pruebas de caja negra se validó el correcto funcionamiento del software, las cuales arrojaron como resultado que luego de tres iteraciones el módulo quedara libre de no conformidades.

La aplicación de la técnica Iadov constató la satisfacción del cliente con la entrega del módulo GOE versión 2.0, reflejándose un alto grado de satisfacción grupal con la solución obtenida para elevar el control de la información.

CONCLUSIONES

El estudio de sistemas los sistemas homólogos, arrojó que ninguno brindaba una solución completa a las particularidades de la planificación de la GOE en el escenario de la Facultad 3, lo cual constituyó motivo para el desarrollo de la investigación realizada. Se decidió desarrollar la implementación de la solución con las herramientas empleadas en la primera versión del módulo: Symfony 2 en su versión 2.3 como marco de trabajo, Bootstrap 3 para el manejo del componente visual y jQuery 1.9 para el tratamiento de errores y manejo del DOM, siguiendo como metodología de desarrollo de software AUP en su variación UCI.

Se identificaron los requisitos funcionales y no funcionales para el módulo, constatándose mediante la aplicación de las métricas de validación de requisitos, que en su totalidad son no ambiguos, comprensibles y correctos.

La aplicación de las métricas del diseño TC, CBO y LCOM, corroboró la existencia de un bajo acoplamiento y alta cohesión en las clases, por lo que se considera que el diseño es correcto, hay una alta reutilización y una correcta asignación de responsabilidades.

La aplicación de los estándares de codificación permitió una implementación responsable y organizada facilitando la obtención del módulo GOE versión 2.0 del SAEF3, además de garantizar la posibilidad de mantenimientos futuros al módulo. Para verificar el correcto funcionamiento de la solución obtenida se aplicaron pruebas de caja negra, donde luego de tres iteraciones la solución quedó libre de no conformidades.

La aplicación de la técnica Iadov permitió medir la satisfacción del cliente respecto a la solución desarrollada, donde los resultados obtenidos se traducen en un elevado agrado por parte del cliente con el módulo en su versión 2.0. La validación de las variables de la investigación corroboró mejoras respecto al control de la información asociada a los procesos de planificación y control de la GOE.

REFERENCIAS BIBLIOGRÁFICAS

AGUILAR, O. L. AND L. A. R. CASTILLO. Desarrollo del Módulo GOE para el Sistema de Administración y Economía de la Facultad 3. UCI, 2015.

AMBLER, S. W. *Modelagem ágil: práticas eficazes para a Programação Extrema e o Processo Unificado*. Edtion ed.: Bookman Editora, 2009. ISBN 8577804135.

APACHE. Apache - openSUSE. In., 2011, vol. 2016.

BEJERANO, K. G. AND F. M. B. MONTES DE OCA. Aplicación Web para el control de la Guardia Estudiantil y la Cuartelería de la Facultad 7. UCI, 2009.

BRAVO, S. M., M. Á. G. CONDE AND F. J. P. GARCÍA. Ingeniería de Software. In.: Universidad de Salamanca. Departamento de Informática y Automática, 2008, vol. 2016.

BROWN, C. V. *IS Management Handbook, Seventh Edition*. Edtion ed.: Taylor & Francis, 1999. ISBN 9780849398209.

BUSCHMANN, F., K. HENNEY AND D. C. SCHMIDT *Pattern-Oriented Software Architecture: A Pattern Language For Distributed Computing*. Edtion ed., 2007. ISBN 8126513004.

CAMPOS, A. L. AND V. M. MÉNDEZ. Sistema de gestión docente metodológica integrado con la plataforma informativa SO3. UCI, 2013.

CHIDAMBER, S. R. AND C. F. KEMERER A Metrics Suite for Object Oriented Design. Software Engineering, IEEE Transactions, 1994.

DÍAZ, G. Y. AND R. Y. FERNÁNDEZ Patrón Modelo-Vista-Controlador. Telemática. Revista Digital de las Tecnologías de la Información y las Comunicaciones, 2012, 11(1).

EMR, S. Jano Seguridad. In., 2008.

FIGUEROA, R. G., C. J. SOLÍS AND A. A. CABRERA Metodologías Tradicionales VS. Metodologías Ágiles. Loja, 2008.

GAMMA, E., R. HELM, R. JOHNSON AND J. VLISSIDES. Patrones de diseño. Elementos de Software orientado a objetos reutilizables. In.: Addison-Wesley, 2002.

GARCIA, A. M. R., G. H. DARIAS AND B. Y. G. PÉREZ Validación de requisitos con expertos funcionales. Informática 2013, 2013.

GUTIÉRREZ, F. R. F. AND A. A. Z. TAPIA. Desarrollo de Sistema Web de Asignación Automática de Mallas de Turnos para Protector Security. In. Chile: Universidad de Talca, 2013.

HETZEL, B. *The Complete Guide to Software Testing*. Edtion ed., 1993. ISBN 978-0471565673.

HUMPHREY, W. S. *A Discipline for Software Engineering*. Edtion ed. Inc. Boston: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©1995 1995. 816 p. ISBN 0201546108

IEEE. 610-1990 - IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries. In. IEEE Xplore Digital Library: IEEE Computer Society, 1990.

JACOBSON, I., G. BOOCH AND J. RUMBAUGH *El Proceso Unificado de Desarrollo de Software*. edited by ADDISON-WESLEY. Edtion ed., 2000.

JETBRAINS. PhpStorm Lightning-smart PHP IDE. In., 2016.

JIMÉNEZ, N. B. AND Y. F. CONCEPCIÓN. Aplicación web para la gestión de la planificación de la guardia obrera estudiantil en la Universidad de las Ciencias Informáticas. UCI, 2012.

JQUERY-FUNDATION. JQuery, write less do more. In., 2016.

KUZMINA, N. V. *Metódicas Investigativas de la actividad pedagógica*. Edtion ed. Leningrado, 1970.

LARMAN, C. *UML y Patrones*. Edtion ed.: Pearson, 1999. ISBN 8420534382.

LARMAN, C. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Edtion ed. Madrid: Person Educación, 2003.

LORENZ, M. AND J. KIDD *Object-Oriented Software Metrics*. Edtion ed., 1994. ISBN 9780131792920.

MATA, D. S. AND A. Q. NAVARRO. Sistema para la Gestión de la Guardia Obrera-Estudiantil en la Facultad 3. UCI, 2013.

MICROSOFT. Revisiones de código y estándares de codificación. In.: Microsoft, 2016, vol. 2016.

OROZCO, E. S., J. ALCANTAR, J. CARRO SUÁREZ, O. FERNANDO CASTELLANOS, et al. *Inteligencia Empresarial. Qué y cómo*. Edtion ed. La Habana: IDICT, 2009. ISBN 978-959-234-070-1.

PAÑEDA, R. J. B. *ADMINISTRACION*. Edtion ed.: MCGRAW-HILL, 2004. ISBN 9789701038239.

PEDRAZA, J. C. Q. La planificación. Contribuciones a la Economía, 2009.

PGADMIN. pgAdmin PostgreSQL Tools. In., 2016, vol. 2016.

POSTGRESQL, E. D. D. D. Manual del usuario de PostgreSQL. T. LOCKHART, 1996.

POTENCIER, F. Symfony. In.: SesionLabs, 2016, vol. 2016.

PRESSMAN, R. *Software Engineering. A Practtitioner´s Approach*. Edtion ed. New York: McGraw-Hill, 2010. ISBN 978-0-07-337 597 -7.

RAE. Diccionario de la lengua española. In.: Real Academia Española, 2014, vol. 2016.

REFSNES-DATA. w3schools.com. In., 2016, vol. 2016.

RUMBAUGH, J., I. JACOBSON AND G. BOOCH *El Lenguaje Unificado de Modelado*. Edtion ed.: Addison Wesley, 2000. ISBN 84-7829-037-0.

SÁNCHEZ, T. R. Metodología de desarrollo para la Actividad productiva de la Universidad de las Ciencias Informáticas. La Habana: 2014.

SOMMERVILLE, I. *Ingeniería del Software*. Edtion ed.: Pearson. Addison Wesley, 2005. ISBN 84-7829-074-5.

SOTO, F. D. M. AND G. E. POMPA. Desarrollo del módulo económico para el Sistema de Administración y Economía de la Facultad 3. Universidad de las Ciencias Informáticas, 2015.

SYMFONY. Symfony. In., 2015.

VISUAL-PARADIGM. Visual Paradigm. In., 2016, vol. 2016.

ANEXOS

Anexo 1: Historias de Usuario

RF 1: Obtener Histórico de las Guardias Planificadas.

	Historia de usuario
Número: 1	Nombre del requisito: Obtener Histórico de las Guardias Planificadas.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 4 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 5 horas
Descripción: El sistema debe brindar la posibilidad al usuario de visualizar el registro de guardias realizadas con un conjunto de detalles como nombres y apellidos, fecha, tipo de cumplimiento, observación, área, posta y turnos realizados por planificación, cuenta con un buscador para realizar más fácil la búsqueda de la información deseada por el usuario. Además, brinda la funcionalidad de realizar una búsqueda exhaustiva mediante el filtrado por un rango de fecha seleccionado por el usuario arrojando el listado del registro de guardias que pertenezcan a dicho intervalo de fechas.	
Observaciones: Para la funcionalidad de búsqueda mediante el rango de fechas de no llenarse ambos campos el sistema no efectuará la operación.	
Prototipo elemental de interfaz gráfica de usuario:	

RF_2: Configurar rotaciones de los turnos en la planificación de la guardia.

Historia de usuario	
Número: 2	Nombre del requisito: Configurar rotaciones de los turnos en la planificación de la guardia.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 10 horas
Riesgo en Desarrollo: Alto	Tiempo Real: 9 horas
Descripción: Permite al usuario establecer la configuración de rotación de los turnos que va a tomar la planificación de la guardia.	
Observaciones: De no introducir ninguna configuración de rotación el sistema planificará sin tener en cuenta dicho parámetro.	
Prototipo elemental de interfaz gráfica de usuario:	

RF_3: Asignar roles de usuarios en el sistema.

Historia de usuario	
Número: 3	Nombre del requisito: Asignar roles de usuarios en el sistema.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Alto	Tiempo Real: 6 horas
Descripción: El sistema debe brindar la posibilidad de asignar roles a cada usuario para el control del acceso a este y en relación con los roles asignados a cada cual se definen niveles de privilegios.	
Observaciones:	

Todo usuario ingresado en el sistema se registrará con rol registrado, el administrador es el único capaz de cambiar el rol a cada usuario .
Prototipo elemental de interfaz gráfica de usuario:

RF_5: Mostrar factibilidad de guardia configurada.

		Historia de usuario
Número: 5	Nombre del requisito: Mostrar factibilidad de guardia configurada.	
Programador: Osvaldo Alejandro Pacheco Morales.		Iteración Asignada: 1
Prioridad: Alta		Tiempo Estimado: 6 horas
Riesgo en Desarrollo: Media		Tiempo Real: 4 horas
Descripción: Permite visualizar un análisis previo a la planificación donde informa si se cuenta con personal suficiente para cubrir todas las postas y los turnos. Además establece una cantidad de días máximos con los que se podrá planificar la guardia.		
Observaciones: No tiene en cuenta las restricciones del personal en el análisis de factibilidad.		
Prototipo elemental de interfaz gráfica de usuario:		

RF_6: Gestionar restricción por grupo.

		Historia de usuario
Número: 6	Nombre del requisito: Gestionar restricción por grupo .	
Programador: Osvaldo Alejandro Pacheco Morales.		Iteración Asignada: 1

Prioridad: Alta	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 14 horas
Descripción: El sistema debe brindar la posibilidad al usuario de gestionar restricción por grupo para un conjunto de personas. La primera vista Gestor de restricciones contendrá un select el cual carga la relación de grupos mediante el consumo de un servicio, después de seleccionado el grupo deseado el usuario tiene la opción de ver la relación de integrantes del grupo mediante la opción Mostrar o puede realizar la operación de Adicionar. La segunda vista Adicionar restricción grupo permitirá al usuario añadir una restricción por fecha al grupo seleccionado.	
Observaciones: Si no se selecciona el grupo en ambas vistas y se llena el campo de fecha el sistema no realizará la operación.	
Prototipo elemental de interfaz gráfica de usuario:	

RF 7: Gestionar restricción del personal

		Historia de usuario
Número: 7	Nombre del requisito: Gestionar restricción del personal.	
Programador: Osvaldo Alejandro Pacheco Morales.		Iteración Asignada: 1
Prioridad: Alta		Tiempo Estimado: 6 horas
Riesgo en Desarrollo: Bajo		Tiempo Real: 6 horas
Descripción: El sistema debe brindar la posibilidad al usuario de gestionar restricciones a los implicados en la planificación de la guardia. La vista Gestionar restricción muestra el listado de personas denominados potencial de guardia, al cual se le podrá insertar		

restricciones según los tipos mostrados en el listado mediante un modal desplegable el cual contiene la opción de adicionar restricción. La vista Adicionar restricción contendrá todos los campos de tipos de restricciones manejados por el sistema y permitirá al usuario añadirlas a la persona en cuestión .
Observaciones: Para desplegar el modal debe seleccionarse la persona a adicionar la restricción.
Prototipo elemental de interfaz gráfica de usuario:

RF 8: Autenticar usuario mediante consumo de servicios.

Historia de usuario	
Número: 8	Nombre del requisito: Autenticar usuario mediante consumo de servicios.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 14 horas
Descripción: El sistema debe brindar la posibilidad de acceso a toda persona con usuario y contraseñas registradas en bases de datos de la universidad mediante el servicio de autenticación correspondiente.	
Observaciones: Si no se posee credenciales en la universidad no tiene acceso al sistema.	
Prototipo elemental de interfaz gráfica de usuario:	

RF 9: Gestionar potencial de la guardia.

Historia de usuario	
Número: 9	Nombre del requisito: Gestionar potencial de la guardia.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 14 horas
Descripción: El sistema carga los datos del potencial de guardia mediante el consumo de varios servicios publicados en la universidad, debido a esto el potencial se carga en dos grupos, estudiantes y trabajadores cada uno con sus respectivas vistas similares entre sí, pero con sus particularidades. En la vista Gestionar potencial de trabajadores, se carga en un select toda la información de las áreas de la uci mediante el consumo de un servicio, se brindan 2 opciones las de mostrar los datos de esa área o la opción de cargar los datos en el sistema.	
Observaciones: Para realizar cualquiera de las 2 operaciones debe estar seleccionada un área en el select.	
Prototipo elemental de interfaz gráfica de usuario:	

RF 10: Gestionar mensajes de notificación del sistema.

Historia de usuario	
Número: 10	Nombre del requisito: Gestionar mensajes de notificación del sistema.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 4 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar los mensajes de notificación manejados por el sistema. La primera vista contiene el listado de notificaciones ya registradas en el sistema según un tipo de notificación el cual no es configurable y brinda la opción de insertar una nueva notificación. Además, esta vista brinda la posibilidad de mostrar los detalles de una notificación mediante un modal desplegado al seleccionar una notificación del listado y en este se brinda la opción de eliminar la notificación y la opción de editarla.

La segunda vista Adicionar notificación permitirá al usuario añadir una nueva notificación al insertar el tipo y el contenido.

La segunda vista Editar notificación permitirá al usuario editar la notificación seleccionada al modificar el tipo y el contenido.

Observaciones:

Si no se llenan los campos no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:

RF 11: Notificar la planificación mediante correo a usuarios registrados en el sistema.

Historia de usuario	
Número: 11	Nombre del requisito: Notificar la planificación mediante correo a usuarios registrados en el sistema.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 9 horas
Descripción: El sistema después de realizada la planificación de la guardia realiza una notificación por correo a usuarios registrados en el sistema donde informa toda la información asociada a esta como área , fecha posta y turno a realizar la guardia.	
Observaciones: Si no se encuentra registrado en el sistema el usuario no se efectuará dicha notificación.	
Prototipo elemental de interfaz gráfica de usuario:	

RF 12: Exportar a Excel la planificación de la guardia.

Historia de usuario	
Número: 12	Nombre del requisito: Exportar a Excel la planificación de la guardia
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 6 horas

Descripción: El sistema en varias vistas permite exportar a formato Excel la información de la planificación generada automáticamente.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

RF_13: Exportar a Excel el registro de cumplimiento.

	Historia de usuario
Número: 13	Nombre del requisito: Exportar a Excel el registro de cumplimiento.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 6 horas
Descripción: El sistema en varias vistas permite exportar a formato Excel la información referente al cumplimiento de la guardia registrado en el sistema.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

RF_14: Gestionar solicitudes de cambio de guardia.

	Historia de usuario
Número: 14	Nombre del requisito: Gestionar solicitudes de cambio de guardia.

Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Media	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 6 horas
Descripción: Permite a los usuarios del sistema realizar permutas entre planes de guardias no realizados. En la primera vista se muestra el listado de planes potenciales a solicitar la permuta y mediante la opción solicitar se realiza la petición a la persona que se le planificó con esa relación, a la cual se le notifica de la solicitud y en la vista Solicitud de permutas tiene la opción de aceptar dicha solicitud o cancelarla.	
Observaciones: No puede realizar notificaciones consigo mismo y si ya tienes registrado el cumplimiento de la guardia no permite realizar permutas a dicho usuario. Si tiene una solicitud en estado pendiente no permite realizar otras solicitudes de permuta.	
Prototipo elemental de interfaz gráfica de usuario:	

RF 15: Intercambiar planificación de guardia entre personas.

	Historia de usuario
Número: 15	Nombre del requisito: Intercambiar planificación de guardia entre personas.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Media.	Tiempo Estimado: 4 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 5 horas

Descripción: Permite al administrador intercambiar la planificación de la guardia entre dos personas. En una primera vista se lista todas las permutas realizadas por este concepto, en ella se da la posibilidad de realizar la permuta a través de la opción Nueva permuta. Además, brinda la posibilidad de mostrar los detalles de una permuta al seleccionarla en el listado desplegando un modal el cual brinda la posibilidad de eliminar el registro de la permuta. En la vista Nueva permuta se seleccionan ambas planificaciones y se realiza el cambio.
Observaciones: Si no se seleccionas amos planes no se realizará la operación.
Prototipo elemental de interfaz gráfica de usuario:

RF 16: Mostrar detalles de la planificación de la guardia.

Historia de usuario	
Número: 16	Nombre del requisito: Mostrar detalles de la planificación de la guardia. .
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 10 horas
Descripción: El muestra un listado de los planes de guardia realizados a traves de la planificación efectuada a los cuales no se les ha registrado cumplimiento.	
Observaciones:	

Prototipo elemental de interfaz gráfica de usuario:

RF 17: Gestionar cumplimiento de la guardia:

Historia de usuario	
Número: 13	Nombre del requisito: Gestionar cumplimiento de la guardia.
Programador: Osvaldo Alejandro Pacheco Morales.	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo	Tiempo Real: 10 horas
Descripción: Permite al administrador del sistema registrar el cumplimiento de las guardias planificadas. En una primera vista se listan todos los planes que no se le han registrado el cumplimiento en la cual se brinda la posibilidad de registrar todos los cumplimientos a la misma vez o de manera individual. Al seleccionar una de las 2 opciones se muestra una vista en la cual se llenan los campos de tipo y observación con las particularidades para cada opción	
Observaciones: Si no se llenan los campos no se realiza la operación.	
Prototipo elemental de interfaz gráfica de usuario:	

RF 18: Notificar incumplimiento de guardia a personal implicado.

		Historia de usuario
Número: 17	Nombre del requisito: Notificar incumplimiento de guardia a personal implicado.	
Programador: Osvaldo Alejandro Pacheco Morales.		Iteración Asignada: 1
Prioridad: Alta		Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo		Tiempo Real: 10 horas
Descripción: El sistema una vez registrado un incumplimiento de guardia a un plan, notifica automáticamente al personal involucrado.		
Observaciones: Para realizar la notificación el personal implicado debe estar registrado en el sistema.		
Prototipo elemental de interfaz gráfica de usuario:		

RF 19: Notificar registro de cumplimiento de la guardia a jefes de área.

		Historia de usuario
Número: 16	Nombre del requisito: Notificar registro de cumplimiento de la guardia a jefes de área.	
Programador: Osvaldo Alejandro Pacheco Morales.		Iteración Asignada: 1
Prioridad: Media		Tiempo Estimado: 8 horas
Riesgo en Desarrollo: Bajo		Tiempo Real: 10 horas
Descripción: El sistema una vez registrado el cumplimiento de los planes de guardia realiza la notificación a los respectivos jefes de área que se encuentran registrados en el		

sistema.
Observaciones: El jefe de área debe estar registrado en el potencial de guarida y en el sistema para que dicha funcionalidad se realiza correctamente.
Prototipo elemental de interfaz gráfica de usuario:

Anexo 2: Prototipos de interfaz de usuarios

Figura 10: Listar restricción. Fuente: (Elaboración propia).

Figura 11: Adicionar restricción. Fuente: (Elaboración propia).

Figura 12: Listar plan guardia. Fuente: (Elaboración propia).

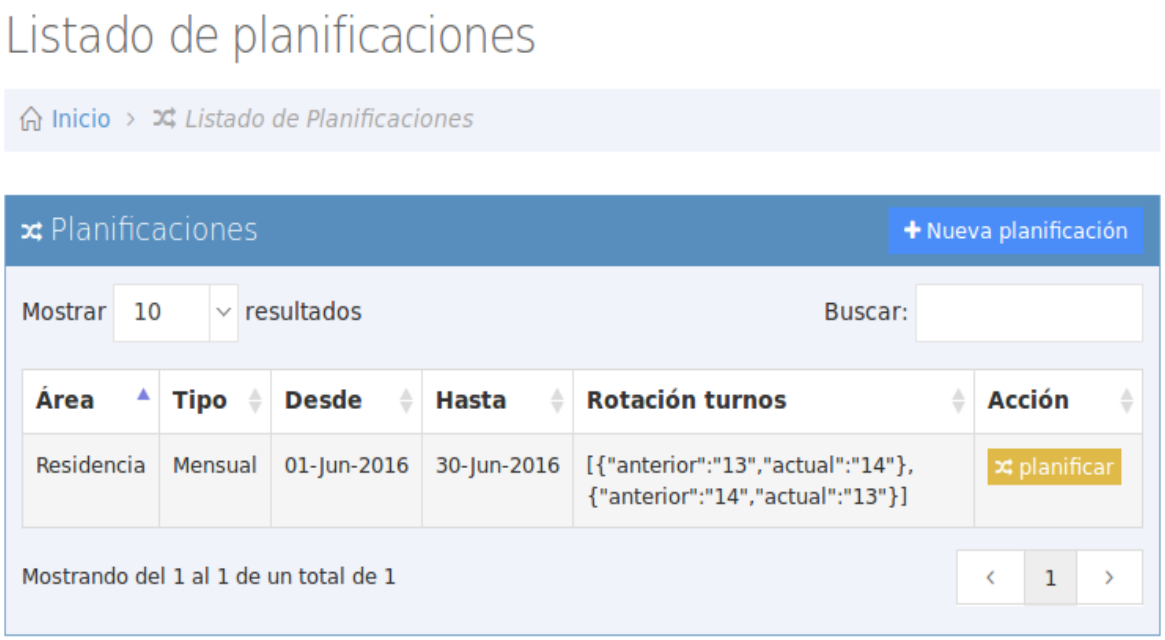

Figura 13: Listar planificación. Fuente: (Elaboración propia).

Figura 14: Listar cumplimiento. Fuente: (Elaboración propia).

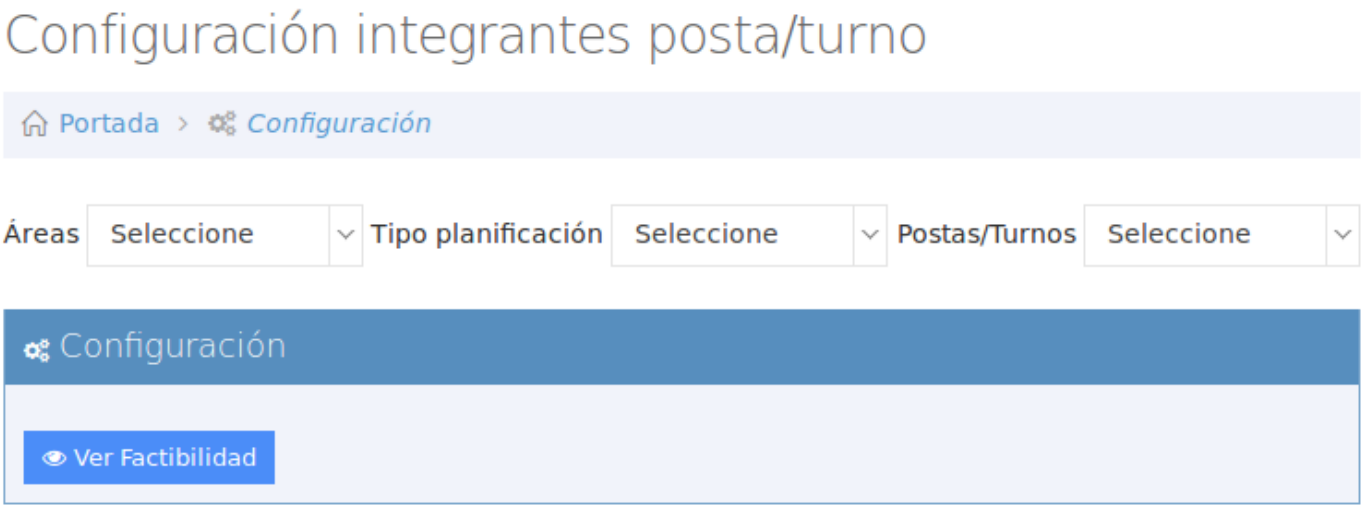

Figura 15: Mostrar factibilidad. Fuente: (Elaboración propia).

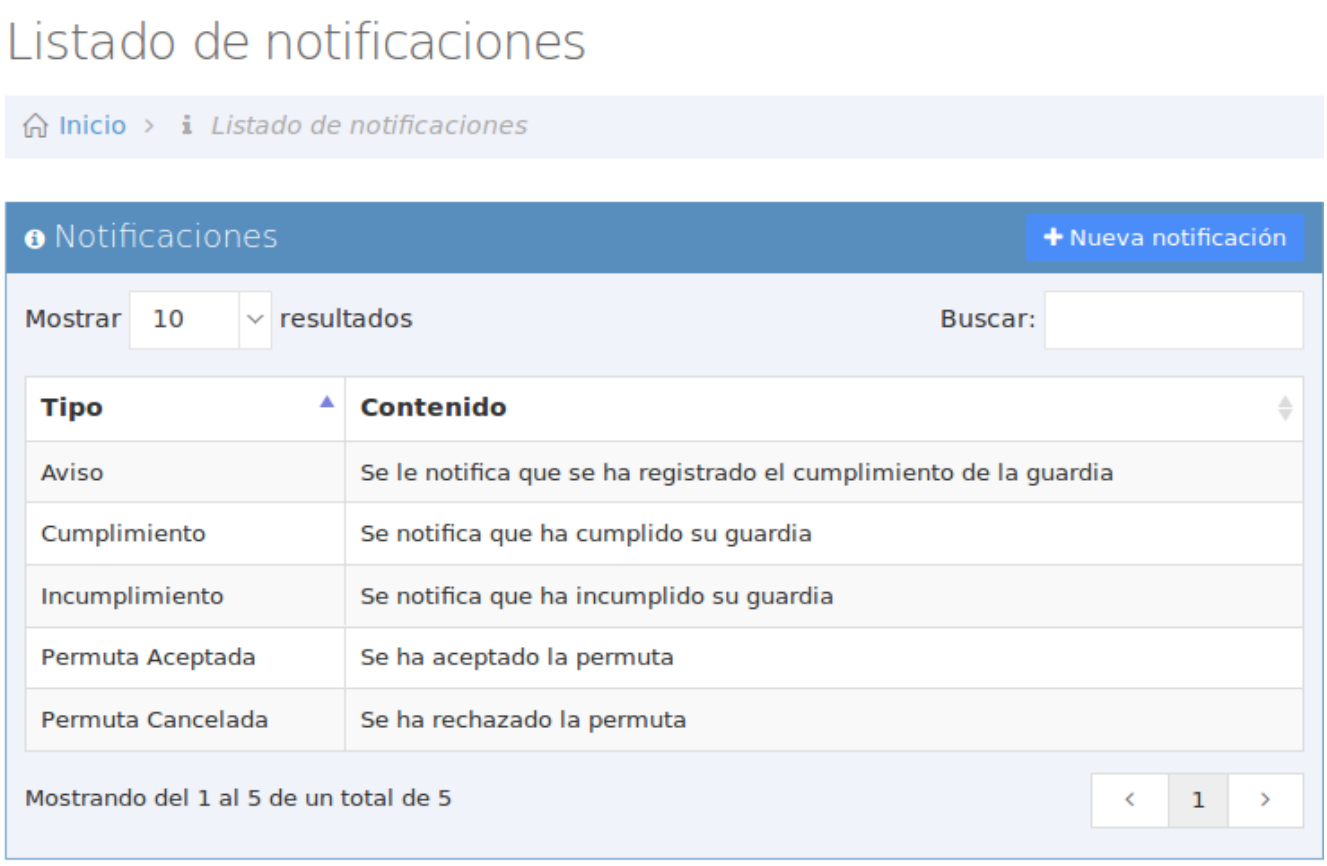

Figura 16: Listar notificaciones. Fuente: (Elaboración propia).

Anexo 3: Diagramas de clases con estereotipos web

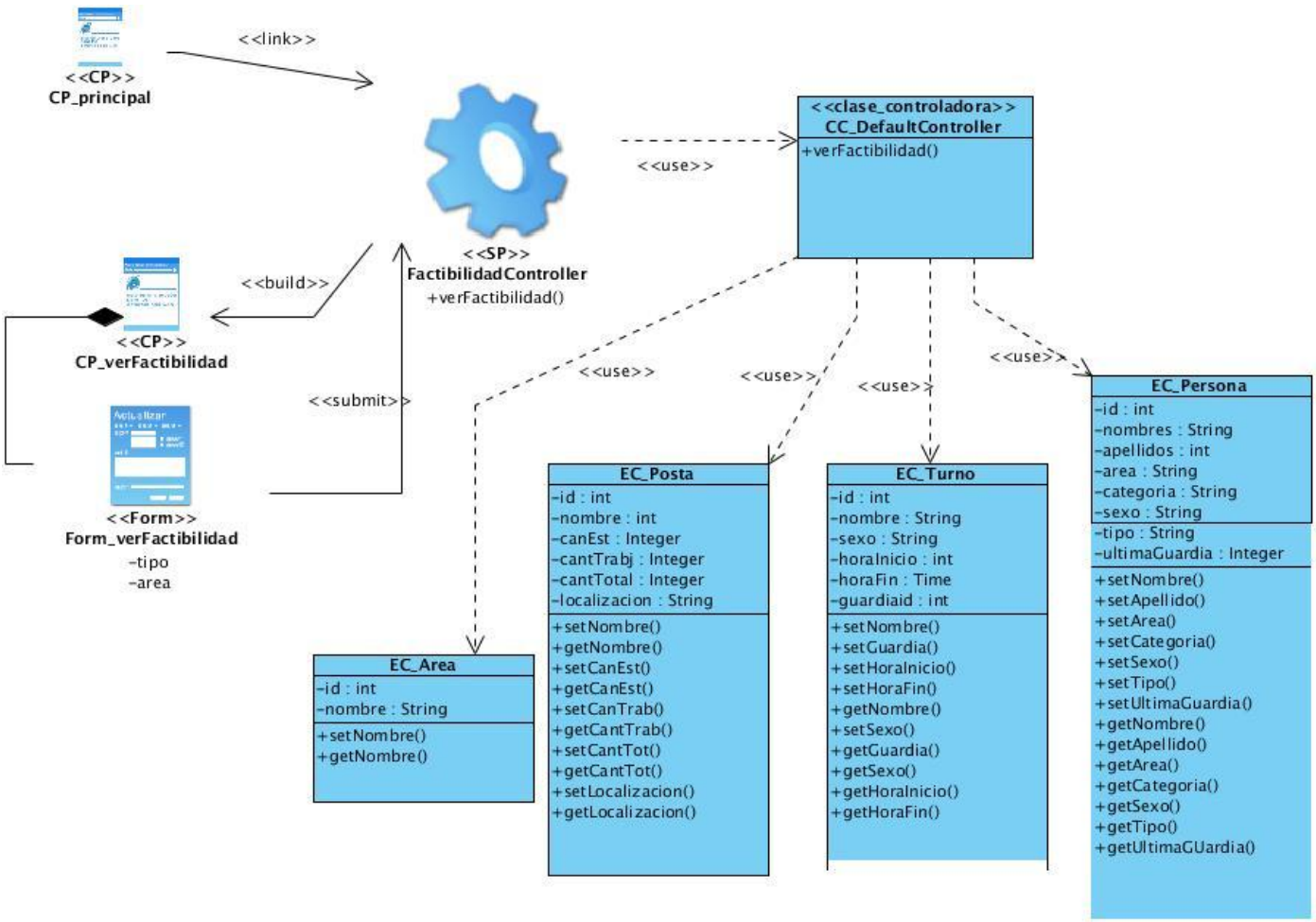

Figura 17: Mostrar factibilidad. Fuente: (Elaboración propia).

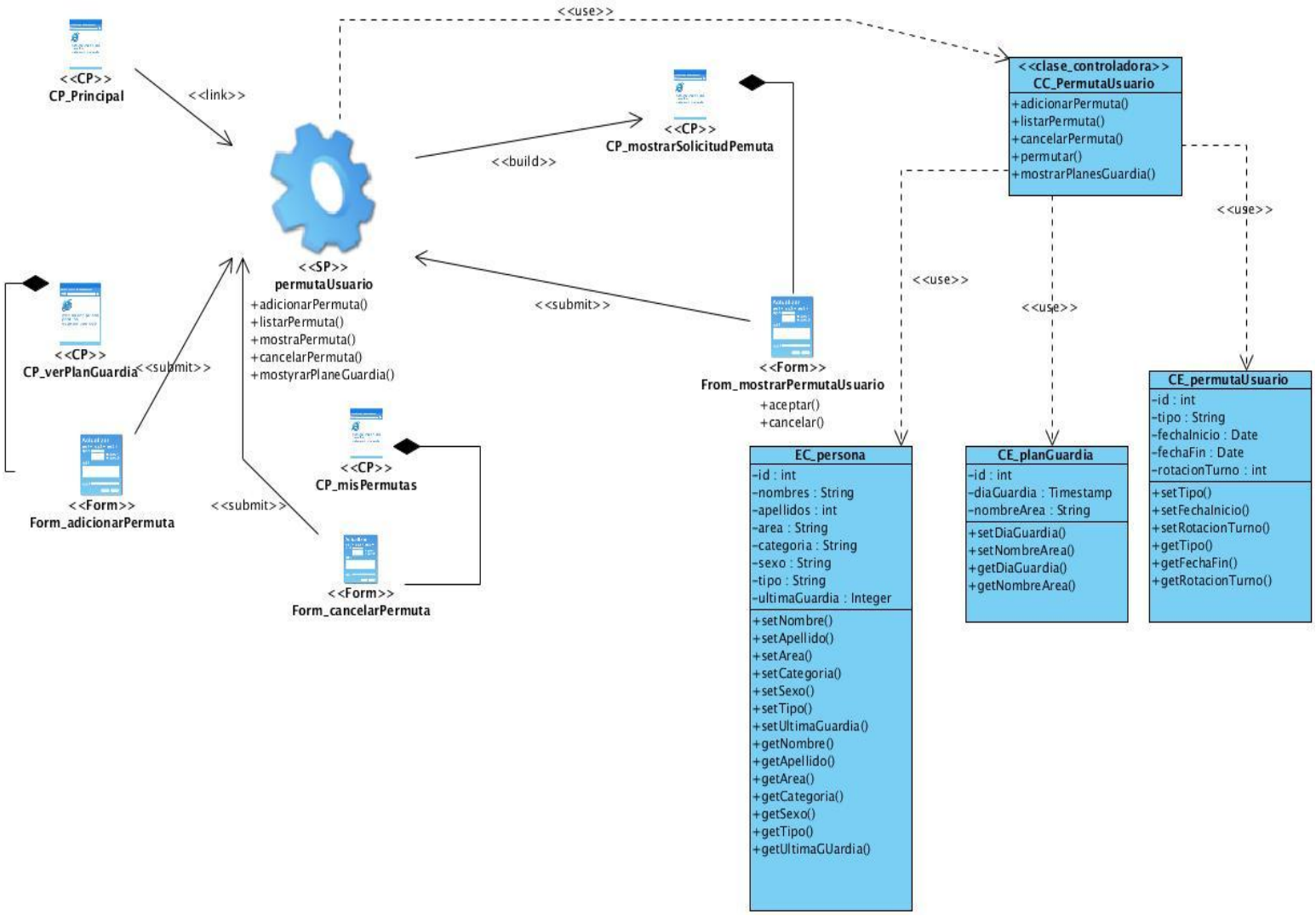

Figura 18: Permuta usuario. Fuente: (Elaboración propia).