Universidad de las Ciencias Informáticas

Facultad 3

Aplicación web para la gestión de la planificación de la guardia obrera estudiantil en la Universidad de las Ciencias Informáticas.

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

Autor(es):

Nardely Jiménez Barreto
Yasser Concepción Fernández

Tutor: Lic. Sarianny Olivares Aguilar
Co-Tutor: Ing. Yenier Figueroa Machado

"Inteligencia más carácter, el objetivo de una verdadera educación"

Dr. Martin Luther King. Jr.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ Nardely Jiménez Barreto

_____ Yasser Concepción Fernández

_____ Lic. Sarianny Olivares Aguilar

_____ Ing. Yenier Figueroa Machado

AGRADECIMIENTOS

A nuestros padres y hermanos, por el amor que han sabido darnos, por las lecciones de vida que nos han enseñado, por saber guiar nuestros pasos durante todo este tiempo que nos íbamos convirtiendo en gente de bien.

A todos los que de una forma u otra han influido en nuestras vidas aportando su pedacito de experiencia en nuestra forma de pensar, vivir y amar.

A todos los que no se cansaron nunca de creer en nosotros, nos apoyaron e hicieron suya esta investigación.

¡Gracias!

DEDICATORIA

“A mis padres y hermano, por saber guiarme y brindarme su confianza y apoyo incondicional con el cual he podido llegar a este día. A todos los que como yo, luchan sin cejar, por hacer realidad sus sueños, por más descabellados que estos puedan llegar a ser.”

Yasser.

“El mayor sueño de una persona es prolongar la vida más allá de la suya propia por lo quiero dedicar este logro a mi hijo Jan Carlos que es el motivo por el que cada día decido seguir adelante, no ha sido mi única opción sino mi mejor decisión. También a mis padres que me han mostrado el camino cuando yo no he sido capaz de verlo y me han dado fuerzas cuando creía no poder. También quiero dedicarlo a mi hermano que fue mi mayor ejemplo y la razón que inicialmente inspiró mis decisiones que hoy están dando frutos.”

Nardely.

RESUMEN

Con la presente investigación se pretende crear una Sistema Informático para la gestión de la planificación de la Guardia Obrera Estudiantil en la Universidad de las Ciencias Informáticas. De esta forma se brinda una posible solución a los actuales problemas que presenta este proceso en dicha Universidad y por consiguiente la falta de control sobre el mismo. Se define como factores que influyen en el cumplimiento de la Guardia la repetición de turnos, días y postas; a su vez se identifican una serie de reglas a tener en cuenta que pueden dificultar o complejizar esta labor cuando es llevada a cabo por una persona. Haciendo uso de un algoritmo de selección estratégica, se logra una planificación donde no se repitan consecutivamente los días, turnos y postas de guardia y se satisfagan las necesidades básicas para el cumplimiento de la misma, tanto para estudiantes y trabajadores involucrados en este proceso.

Para el desarrollo de la solución propuesta, se lleva a cabo un análisis comparativo de Herramientas y Metodologías de desarrollo de las cuales se seleccionan las que cumplen con un grupo de indicadores previamente definidos. A su vez se hace uso de buenas prácticas de la programación con el uso de patrones y métricas que faciliten y simplifiquen la implementación de la solución propuesta teniendo en cuenta el FrameWork de desarrollo seleccionado.

PALABRAS CLAVES

Evaluación, Gestión, Guardia, Planificación, Postas, Turnos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Se puede decir que el origen y evolución del término planificación, es producto de las necesidades del mundo en que vivimos y las organizaciones que lo rigen. En la actualidad planificar es un proceso complejo que busca articular lo que se tiene, con lo que se quiere y debe lograr dentro de una organización, involucrando toda una estructura de análisis y búsqueda de soluciones. Las tendencias sobre este proceso se orientan a concepciones estratégicas y fórmulas ampliamente participativas que se enriquecen en cada organización, con el fin de dar dirección y sentido a los procesos que quieren lograr, aprovechando sus fortalezas internas.

En ámbitos como las tecnologías de la información y comunicaciones (TIC) se ha integrado la planificación para lograr una mejor organización y agilización de los procesos, creándose una tendencia a la utilización de sistemas informáticos capaces de realizar planificaciones. Estos sistemas ayudan en la posterior toma de decisiones, al eficaz manejo de la información y la gestión de recursos tecnológicos y humanos necesarios

En Cuba, la mayoría de las instituciones hace uso de la planeación ya sea en la educación con la planificación de horarios y actividades, en la salud con la planificación de turnos o en la economía con planeación de recursos. Una de las áreas en que se aplica la planificación es en la guardia obrera, tal y como se establece en la Gaceta Oficial #021 del 2007 (1). La Universidad de las Ciencias Informáticas (UCI) es uno de estos centros educacionales donde se lleva a acabo dicho proceso al cual se le incluye la guardia estudiantil. Este proceso se constituye como Guardia Obrera Estudiantil (GOE).

A medida que la UCI ha ganado madurez en los procesos que se desarrollan en sus áreas, con el tiempo han sido sometidos a una revisión para su mejora. En el curso 2012-2013 la GOE como proceso fundamental de la institución para el cuidado y custodia de sus recursos y bienes materiales, ha transitado por varios cambios en su concesión, a pesar de todos los esfuerzos por mejorar los índices de cumplimiento, se identifican algunas deficiencias asociadas a la gestión de la planificación lo que trae consigo:

- Planificación ineficiente, ya que a consideración de los autores e involucrados en el proceso y atendiendo a la revisión de la documentación consultada, este se realiza de manera empírica, la socialización de la información desde el planificador al usuario final

no llega con el tiempo requerido y el margen de error de coincidencias en postas, tunos y horarios provoca inconformidades en los involucrados.

- Limitado acceso a la información generada en el proceso de planificación, está dado principalmente por el deterioro y la pérdida de la información que está almacenada en archivos impresos o múltiples documentos de ofimática. Existe además una centralización de la información en el área responsable de la planificación, lo que provoca un escaso conocimiento de los resultados del proceso.

- Afectación en la consolidación y confiabilidad de la información, principalmente por la no obtención de reportes estadísticos en función de un comportamiento determinado.

Lo antes expuesto constituye la problemática que origina esta investigación de la cual se define el siguiente **problema a resolver**: ¿Cómo contribuir al seguimiento de la Guardia Obrera Estudiantil de la Universidad de las Ciencias Informáticas para mejorar la gestión de la planificación?

Teniendo en cuenta el problema identificado, el **objeto de estudio** es: Gestión de la planificación.

Mientras que el **campo de acción** lo constituye: Gestión de la planificación de la Guardia Obrera Estudiantil de la Universidad de las Ciencias Informáticas.

En correspondencia con el problema a resolver se plantea como **objetivo general**: Desarrollar una aplicación web que mejore la gestión de la planificación de la de la Guardia Obrera Estudiantil de la Universidad de las Ciencias Informáticas.

Se toma como idea a defender que si se desarrolla una aplicación web para contribuir al seguimiento de la Guardia Obrera Estudiantil, entonces se mejorará la gestión de la planificación de dicho proceso.

Para darle cumplimiento al objetivo general propuesto se han definido los siguientes **objetivos específicos**:

Elaborar el marco teórico de la investigación mediante un estudio del estado del arte de los procesos relacionados con la Guardia Obrera Estudiantil, gestión de la planificación, metodologías de desarrollo, técnicas y herramientas de software.

Realizar el análisis y diseño de la aplicación web, para la gestión de la planificación de la guardia obrera estudiantil, mediante la metodología de desarrollo de software establecida.

Generar el código de la aplicación web para la gestión de la planificación de la guardia obrera estudiantil, mediante las herramientas y tecnologías seleccionadas.

Validar la solución propuesta mediante pruebas de software y el cumplimiento de los objetivos de la investigación.

Para darle cumplimiento a **los objetivos específicos** de la investigación se traza las siguientes **tareas de la investigación**:

- Estudio del estado del arte de sistemas de gestión enfocados a la planificación.
- Estudio del proceso de desarrollo de software de sistemas de planificación.
- Estudio del proceso Planificación de la Guardia Obrera Estudiantil.
- Elaboración en compañía del cliente de las historias de usuarios.
- Definición en compañía del cliente el plan de iteraciones y el plan de entrega.
- Elaboración de las Tarjetas CRC.
- Definición de las tareas de programación por cada historia de usuario en cada iteración.
- Implementación de las funcionalidades asociadas a cada historia de usuario.
- Realización del diseño de casos de prueba asociados a las funcionalidades implementadas.
- Ejecución de pruebas de software al código fuente obtenido en la implementación de las diferentes funcionalidades.
- Validación de los resultados obtenidos a través de pruebas de caja negra asociados a las diferentes funcionalidades.
- Validación de la propuesta de solución de la investigación frente al problema planteado.

Cumpliendo con cada **objetivo específico** que tributa al cumplimiento del **objetivo general** se tiene como posible **resultado** un sistema funcional para el control y la ejecución del proceso planificación y evaluación de la Guardia Obrera y Estudiantil en la UCI.

Para dar respuesta a las tareas planteadas con anterioridad se proponen **métodos científicos** entre los que se encuentran los siguientes:

Teóricos:

La conformación de una teoría que explique el objeto que se estudia presupone modelar dicho objeto, es decir, abstraer un conjunto de características y relaciones de ese objeto, que explique los fenómenos y hechos que se investigan (2).

Método histórico-lógico: favoreció el estudio de los antecedentes, el desarrollo, las regularidades y tendencias actuales de gestión, planificación, Guardia Obrera Estudiantil, software existentes en el mundo en Cuba y en la UCI, así como las principales metodologías de desarrollo de software.

Método dialéctico: se aplica para analizar las deficiencias de la GOE en la UCI y determinar los posibles cambios para su mejora e informatización.

Analítico - sintético: facilitó la determinación de aspectos generales relacionados con los principales conceptos estudiados, lo que contribuyó también a la sistematización de la información sobre el tema y la selección de los aspectos esenciales para la elaboración del estado del Arte.

Empíricos:

Entrevista: Se aplica a Vicedecanos de Extensión y Residencia, estudiantes, trabajadores y especialistas del área de Seguridad y Protección con el fin de comprobar cómo se realiza la planificación de la GOE y las principales deficiencias que afectan este proceso en la UCI. Además se obtienen con la entrevista información de las principales necesidades del cliente acerca del negocio y los requerimientos para elaborar la propuesta.

Método de Medición: se aplica en la observación del tiempo en que se ejecuta por los administrativos encargados de llevar a cabo este proceso con el fin de medir el tiempo que se emplea en realizar la planificación completa de cada área y así poder contribuir a minimizar

este indicador en la solución propuesta, **Este trabajo ha sido organizado en tres capítulos de la siguiente manera:**

El Capítulo 1: Fundamentación Teórica comprende el estado del arte, que incluye un estudio de sistemas existentes para la gestión y planificación de procesos en el ámbito nacional e internacional y se enuncian los principales conceptos relacionados con el tema. Además se realiza un estudio comparativo de principales herramientas, plataformas de desarrollo y metodología usadas donde se seleccionan y describen las más favorables para darle solución al problema planteado. Se incluyen también un estudio referente a los diversos patrones que son empleados en el desarrollo de sistemas informáticos.

El Capítulo 2: Planificación, Diseño e Implementación maneja elementos relacionados con la descripción de las funcionalidades del sistema a implementar así como el diseño del mismo. Muestra las relaciones de los diferentes artefactos generados durante el proceso de implementación teniendo en cuenta la metodología que será definida.

El Capítulo 3: Validación de los Resultados muestra de los resultados de las pruebas realizadas al sistema informático obtenido como resultado de la investigación. El análisis de estos resultados valida la solución propuesta por la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El presente capítulo aborda los elementos que permitieron realizar una correcta caracterización del sistema, sustentados en un estudio de los conceptos fundamentales a los que está relacionado. Se incluyen la tendencia del desarrollo web actual en el mundo, el país y particularmente en la UCI relacionados con el sistema a desarrollar. Se ofrece un estudio de las principales herramientas, plataforma de desarrollo y metodología de desarrollo de software que son sugeridas. Se estudian los diferentes patrones empleados en el desarrollo de sistemas informáticos y finalmente se hace alusión a las pruebas para evaluar la calidad de las disciplinas descritas en dicho capítulo.

1.2. Conceptos

Entre los principales conceptos usados en la investigación se encuentran los siguientes:

1.2.1. Definición de procesos.

Existe una variedad de autores que han definido lo que es un proceso, esto ha contribuido a una evolución de dicho concepto en el tiempo. Podemos observar esta tendencia analizando algunos de estos autores.

Gabriel Pall, define un proceso como “la organización lógica de personas, materiales, energía, equipamiento e información en actividades de trabajo diseñadas para producir un resultado final requerido” (3). También veremos que Miguel Heras se refiere al proceso como “el conjunto de actividades secuenciales que realizan una transformación de una serie de inputs (material, mano de obra, capital, información) en los outputs deseados (bienes y/o servicios) añadiendo valor” (4). Cabrearas también coincide con Heras acerca de ver la información como una posible entrada de un proceso (5). Mientras que Cooper puntualiza que las salidas también pueden ser información (6).Otras definiciones interesantes de procesos han sido dadas por Juran que lo describe como “Una serie de acciones sistemáticas dirigidas al logro de un objetivo previamente definido” (7).

Podemos concluir que los autores antes mencionados convergen en un mismo sentido, por lo que se asume por los autores de la investigación, como proceso: medio donde se modifica una

entrada ordenada de recursos sobre la cual actúan una serie de acciones que la transforman en función de un objetivo final.

1.2.2. Gestión de procesos.

La **gestión** por sí misma, no es más que la coordinación organizada de los recursos disponibles con el fin de conseguir un objetivo. Teniendo esto en cuenta podemos decir que la gestión de procesos es la forma de gestionar organizadamente las relaciones de los procesos entre ellos y con las demás áreas y el entorno.

La **gestión basada en procesos** se puede definir como la sistemática para identificar y documentar los procesos de una organización, aplicando mediciones de su eficiencia y eficacia y estableciendo planes de mejora continua. A través de esta se puede determinar los procesos que necesitan ser mejorados o rediseñados, estableciendo prioridades y hace posible la comprensión de los procesos de la organización, de sus fortalezas y debilidades.

Un Sistema de Gestión, por tanto, ayuda a una organización a establecer las metodologías, las responsabilidades, los recursos, las actividades que le permitan una gestión orientada hacia la obtención de esos “buenos resultados” que desea, o lo que es lo mismo, la obtención de los objetivos establecidos.

Dentro de la gestión de procesos también podemos encontrar los **indicadores**, los cuales permiten determinar en qué medida de ejecución se encuentra el plan; estos constituyen una herramienta para controlar los resultados por lo que es importante su correcta definición y determinar los que ejercen mayor influencia en los resultados de la actividad a analizar. Estos por definición deben de ser medibles (8) (9).

1.2.3. Planificación.

El tener un conocimiento claro de lo que es planificación, permitirá orientar esta definición a los procesos. Hasta este momento se tiene bien definido que es un proceso y en que consiste su gestión, ahora veremos su relación con la planeación. Según Lawrence Gitman (10) la planeación es la selección demisiones y objetivos, y las estrategias, políticas, programas y procedimientos para lograrlos; es también la toma de decisiones; o selección de un curso de acción entre varias alternativas.

Según este autor podemos ver la planeación como un conjunto de procedimientos en función de alcanzar un objetivo definido, para ello se vale de múltiples variables que definen la toma de decisiones. En otro punto de vista, Sainz de Vicuñ (11) la describe como “un ejercicio de análisis y reflexión en el que intervienen múltiples factores, no solo de índole económico, sino también de índole social, política ambiental, que deben tenerse muy presentes”.

La planeación requiere también de tiempo, técnica y organización, por lo que las TIC´s nos permiten optimizar estas variables a la hora de asignar recursos para la realización de este proceso, además disminuye el nivel de incertidumbre y la ocurrencia de errores introducidos por el accionar del hombre. Fernando A. Gabaldon (12) describe tres características de la planificación y las cuales son adoptadas en esta investigación.

Prospección al futuro: Una dimensión importante de la planificación es el tiempo. No obstante, lo que realmente interesa es el futuro, el pasado es importante solo en la medida en que sirve para proyectar el futuro.

Sistemas de decisiones: La toma de decisiones se constituye un elemento fundamental del proceso, dado que hay que definir lo que se va a hacer, aquí es preciso escoger una entre varias alternativas.

El riesgo: La planificación conlleva riesgo, se comprometen los recursos de la organización en acciones cuyos resultados no se conocen con certeza. Obviamente la planificación no elimina el “riesgo” de hecho lo implica. Lo que sí es cierto es que la planificación contribuye a reducir el riesgo de una decisión en la medida en que disminuye el desconocimiento sobre los factores que la influencian.

Podemos entonces concluir que la planeación es un proceso en sí misma, con entradas, acciones y salidas, lo que facilita su aplicación a los procesos de una entidad.

En un sistema informático se pueden manejar sus características fundamentales, el almacenamiento histórico de los datos generados, permite su fácil acceso para su posterior uso dentro del mismo flujo del proceso. La toma de decisiones se limita a una propuesta simple y sin intervención de variables que aumenten la complejidad de la misma para que mediante el accionar del hombre se pueda ajustar y realizar una nueva propuesta. De este modo el riesgo es mínimo aunque no deja de estar presente pero no deja de ser directamente proporcional a la

cantidad de variables que intervengan durante la ejecución del proceso de planeación, así tenemos que mientras más factores influyan en la misma mayor será el riesgo a tener en cuenta.

2.1.1. Información:

La información es un recurso que supone una inversión y puede ser explotada y abastecerse. La información es un recurso estratégico pues significa conocimiento y control, además es una poderosa arma en la toma de decisiones a cualquier nivel. Es también un arma para el desarrollo de los pueblos para su crecimiento y para la realización personal.

2.1.2. Gestión de la información:

Conjunto de técnicas utilizadas para amentar la productividad del trabajo intelectual mediante una adecuada gestión de las necesidades de información que existan. Es todo lo que se refiere a la obtención de información adecuada, para la persona adecuada en el tiempo y lugar adecuados (13).

La gestión de la información se puede definir como el conjunto de actividades realizadas con el fin de controlar, almacenar y, posteriormente, recuperar adecuadamente la información producida, recibida o retenida por cualquier organización en el desarrollo de sus actividades.

1.3. Escenario de proceso.

La planificación de la GOE es un proceso clave en la UCI, de su calidad depende prevenir, detectar, retardar y neutralizar la ocurrencia de amenazas, así como reducir los niveles de riesgos que puedan dar origen a hechos y actividades delictivas de carácter común o contrarrevolucionario. Esta tarea tiene lugar en diferentes áreas de la UCI y en ella intervienen tanto estudiantes como trabajadores. Para esto se realiza una planificación de guardia para un periodo de tiempo determinado.

El encargado de planificar la guardia tiene la misión de gestionar la planificación teniendo en cuenta personal disponible, horarios y postas distribuidas en área de la universidad. Una vez realizado esto se realizan ajustes según se estimen convenientes y se da a conocer la información a los involucrados en el proceso. Para ello es necesario emplear gran cantidad de tiempo en distribuir correctamente a cada uno de los involucrados, teniendo en cuenta como elemento fundamental que no haya coincidencias en el otorgamiento de turnos y días.

Una vez planificada la guardia se lleva a cabo el proceso de control. Para esta tarea se destina a un grupo determinado de controladores jefes de la guardia. Después del chequeo se emite una evaluación la cual se evalúa de manera cualitativa con una ponderación de: Bien, Regular o Mal y los no evaluados (N/E), se pueden reportar incidencias en caso de detectar anomalías. Las evaluaciones emitidas pueden ser modificadas por el encargado de planificar, además que se evalúa al jefe de la guardia según su desempeño gestión durante el proceso.

En el caso de los trabajadores, según la evaluación efectuada se analiza en los consejos de direcciones de sus áreas los evaluados de M, para el análisis de la pérdida de requisitos para el otorgamiento del pago adicional.

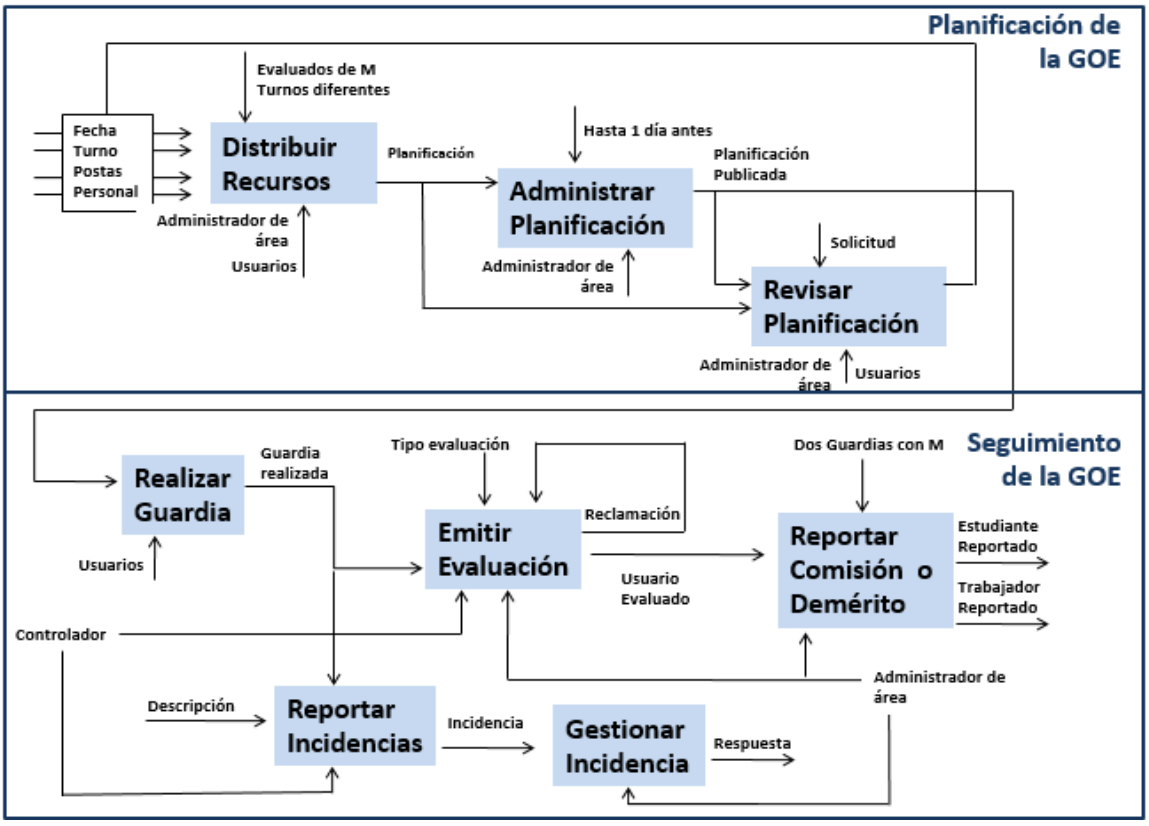


Figura 1 Proceso de la GOE

1.4. Estado del arte

Según la investigación realizada sobre la existencia de otros sistemas informáticos en el campo de la planificación, podemos mencionar algunos desarrollados en otros países y en Cuba. Para

ello se toma una muestra intencionada, a la cual se le definen una serie de indicadores para su selección y comparación:

- **Software Libre:** abarata los costos.
- **Acceso al código fuente**: brinda mayor flexibilidad y seguridad.
- **Documentación Técnica**: mejor entendimiento de la aplicación.
- **Adaptable a la GOE en la UCI**: Capacidad para aplicarlo en la GOE de la UCI.
- **Soporte**: retroalimentación y actualización.
- **Multiplataforma**: Aumenta la posibilidad de uso.

1) **Planifica**: Este software organiza las actividades escolares de tu clase, organizar y planificar sus tareas en el ámbito profesional escolar, también permite exportar sus planificaciones a Word, mediante atractivas plantillas (14). Se encuentra sujeta a una licencia GPL1 y una pobre comunidad de colaboradores.

2) **Kit de Control de Rondas**: Es un completo Sistema que le permitirá lograr con efectividad, hacer el Control de Recorridos de cualquier persona de Seguridad y de todo trabajo rutinario o mantenimiento, logrando obtener luego, completos informes que permitirán evaluar si el recorrido se ha cumplido, en qué fecha y a qué hora estuvo esa persona presente en cada punto de control que se haya establecido (15).

3) **Gestor Web para el control de la Guardia Obrera:** Aplicación desarrollada en la UCI, en la Facultad 8 la cual actualmente no está en funcionamiento. Este sistema tiene en cuenta características específicas de la facultad donde fue creado, no se rige por las nuevas normas establecidas para la realización de la GOE, ya que esta planificación es solo para los trabajadores de la UCI sin tener en cuenta a los estudiantes. Genera reportes referentes al cumplimiento o no de la guardia, aunque estos pueden ser generados en un periodo de tiempo específico (semanal, mensual), no son los únicos reportes necesarios en un proceso como este. La imposibilidad de acceder al código fuente de esta aplicación web imposibilita la reutilización o adaptación de la misma para poder dar solución a la problemática planteada (16).

[1]**GPL**: Licencia Pública General de GNU, garantiza la libertad de usar, estudiar, compartir y modificar el software.

4) **Módulo para la Gestión de la Residencia Estudiantil de la Facultad8**: El sistema de Gestión de Información de la Facultad 8 (SGIF 8), es un sistema para gestionar toda la información referente a la facultad 8, posee 6 módulos, de estos uno es para gestionar gran parte de la información no docente que se lleva a cabo en esta facultad y vinculadas a la residencia estudiantil (17).

5) **Sistema de gestión de información del Área de Extensión Universitaria FAC2:** Entre los procesos de este sistema se encuentra la gestión de la guardia estudiantil, esta está habilitada con las características específicas de la facultad 2 de la UCI, no tiene en cuenta las nuevas regulaciones establecidas para GOE, y no planifica guardia para los trabajadores de la UCI. La imposibilidad de acceder al código fuente de esta aplicación web imposibilita la reutilización o adaptación de la misma para poder dar solución a la problemática planteada (18).

6) **Sistema de Guardia Estudiantil (SiGEst):** Aplicación Web la cual estuvo en uso durante el curso 2010-2011 en la facultad 3 de la UCI. La misma hacía uso de un método Bayesiano para planificar la guardia estudiantil y podía generar reportes. La probabilidad de repetir los turnos haciendo uso de esta aplicación era de un 10% y reducía el tiempo de ejecución de este proceso (realizar la planificación de un mes) a solo 5 min. Esta era una aplicación meramente administrativa aunque contaba con una vista para el uso usuarios, con posibilidad de ver sus guardias y evaluaciones.

Selección

A continuación se procede a comparar mediante indicadores establecidos las principales características de los sistemas informáticos estudiados y la posibilidad de crear una nueva aplicación. Se establece una ponderación según las necesidades del proyecto (se pondera de 0 a 1), mientras mayor sea el puntaje mayor sería la importancia que tiene ese indicador para el proyecto y mientras más indicadores de importancia tenga el sistema analizado así será de favorable su uso:

- **Software Libre**: 0.20
- **Acceso al código fuente**: 0.20
- **Documentación Técnica**: 0.10
- **Adaptable a la GOE en la UCI**: 0.30
- **Soporte**:0.10
- **Multiplataforma**: 0.10

Software / Indicadores	#1	#2	#3	#4	#5	#6
Software Libre (no tiene licencia comercial)	0.20	0	0	0.20	0.20	0.20
Acceso al código fuente	0.30	0	0	0	0	0.20
Documentación Técnica	0.10	0.10	0.10	0.10	0.10	0.10
Adaptable a la GOE en la UCI	0	0	0	0	0.30	0.30
Soporte	0.10	0.10	0.10	0	0	0.10
Multiplataforma	0	0.10	0	0.10	0.10	0.10
Total	**0.70**	**0.30**	**0.20**	**0.40**	**0.70**	**1**

Tabla 1 Comparación de sistemas informáticos estudiados.

Se concluye que la aplicación #6 cuenta con documentación abundante y acceso al código fuente; es posible su adaptación a las nuevas condiciones establecidas para la GOE en la UCI. Es multiplataforma por lo que su despliegue y uso puede ser masivo por parte de todos los usuarios que actúan en el sistema, también se cuenta con la experiencia del equipo de desarrollo el cual es parte de los autores de esta investigación.

A pesar de esto es necesario tener en cuenta algunas desventajas que presenta esta solución y por lo cual se decide realizar una nueva versión de la misma.

Desventajas:

- Vulnerable a los ataques de inyección SQL[2] y Cross-site scripting[3].
- Limitado alcance y adaptabilidad a los cambios.
- No se contempla la planificación de Trabajadores.
- No cumple con las nuevas características de la GOE.
- Planificación Generada inexacta y con un margen de error del 10% de los usuarios planificados.

[2]**Inyección SQL**: es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar consultas a una base de datos.
[3]**Cross-site scripting:** es un tipo de inseguridad informática o agujero de seguridad típico de las aplicaciones Web, que permite a una tercera parte inyectar en páginas web vistas por el usuario código JavaScript o en otro lenguaje script similar.

- Imposibilidad de comunicación con otras plataformas de gestión de la UCI (DataFEU[4] y SOE[5]).

1.5. Propuesta de diseño.

1.5.1. Aplicaciones desktop vs aplicaciones web.

Se realizó un estudio de factibilidad de implementación de la aplicación con enfoque Web o desktop. Las aplicaciones desktop se instalan en máquinas individuales las cuales son las responsables de la ejecución y los resultados que se muestran en la pantalla. Sin embargo las aplicaciones Web son emplazadas en una sola máquina que desempeña el papel de servidor, y son ejecutadas a través de la conexión con un navegador Web, quien se encarga de transportar las solicitudes del cliente y las respuestas del servidor.

Plataforma seleccionada (Aplicación Web)

Se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es un software que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, etc.) en la que se confía la ejecución al navegador. Las aplicaciones web deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, y otros sistemas operativos, la aplicación web se escribe una vez y se ejecuta igual en todas partes.

Por lo general una aplicación web está estructurada como una aplicación de tres-capas. En su forma más común, el navegador web ofrece la primera capa y un motor capaz de usar alguna tecnología web dinámica (ejemplo: PHP) constituye la capa de en medio. Por último, una base de datos constituye la tercera y última capa.

Ventajas

- Acceso ubicuo, sin necesidad de distribución e, idealmente, con pocos requerimientos técnicos. Datos centralizados y fácil integración de datos de múltiples fuentes.
- Permiten el desarrollo de comunidades que dan valor a las aplicaciones (software social).

[4] **DataFEU**: Sistema para la gestión de datos de los estudiantes.
[5] **SOE**: Plataforma Informativa de la Facultad 3.

- Una empresa puede migrar de sistema operativo o cambiar el Hardware libremente sin afectar el funcionamiento de las aplicaciones de servidor.
- No se requieren complicadas combinaciones de Hardware/Software para utilizar estas aplicaciones. Solo un computador con un buen navegador Web.
- Actualizar o hacer cambios en el Software es sencillo y sin riesgos de incompatibilidades. Existe solo una versión en el servidor lo que implica que no hay que distribuirla entre los demás computadores. El proceso es rápido y limpio.
- Se facilita el trabajo a distancia. Se puede trabajar desde cualquier PC o computador portátil con conexión a Internet o a una red interna o privada.
- Al funcionar en un navegador, se requiere un conocimiento básico de informática para utilizar una aplicación Web.

1.5.2. Marco de trabajo.

En la siguiente tabla, dado que los datos están medidos en una categoría ordinal, se calcula la Mediana (según los indicadores determinados) de los FrameWork estudiados (19).

Escala: **Alto**: 3 / **Medio**: 2 / **Bajo**: 1

	Cake PHP	**Symfony**	**Zend FrameWork**	**CodeIgniter**
Accionar de la comunidad	Medio	Alto	Medio	Medio
Adaptabilidad	Medio	Alto	Alto	Medio
Flexibilidad	Medio	Alto	Medio	Alto
Experiencia del equipo de desarrollo	Bajo	Alto	Bajo	Bajo
Disponibilidad de documentación	Media	Alta	Alta	Alta
Ofrece soporte	Bajo	Alta	Media	Bajo
Actividad en las funcionalidades y características	Bajo	Alto	Alto	Alto
Mediana	**Medio**	**Alto**	**Medio**	**Medio**

Tabla 2 Comparación entre FrameWork estudiados

Calculo de la mediana (20): $\boldsymbol{Me = X(n+1)/2}$

Me= Mediana

n= Total de elementos

X= Posición que ocupa la mediana.

Teniendo en cuenta lo anterior. Se evidencia que Symfony es el de mayor Potencial de uso[6] tiene en comparación con el resto de los FrameWork estudiados, con un alto potencial para ser seleccionado.

FrameWork seleccionado (Symfony 2)

"Symfony es un completo marco de trabajo diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Symfony2 está construido para cumplir los principios bases de crear herramientas que nos permitan desarrollar y construir rápidamente robustas aplicaciones. Symfony está construido sobre la idea de las mejores tecnologías" (21).

Se escoge además este marco de trabajo por su abundante documentación, el potencial de trabajo que presenta para el desarrollo de aplicaciones web de forma rápida, robusta y segura. Es uno de los marcos de trabajo más maduros en cuanto a desarrollo web, que incluye varios bundles[7] de terceros que constantemente están mejorando temas de seguridad, rendimiento y otras características de todas las versiones.

Entre otras características que se conocen se encuentran:

- Versátil porque está basado en componentes
- Útil porque soluciona los retos de la programación web.
- Buenas prácticas porque coge ideas de otros FrameWork.
- Flexible pues cada programador puede usar lo que desee.
- Rendimiento pues exige usar como mínimo PHP 5.3, diferentes archivos de configuración que al final se ejecutan en PHP.
- Documentación
- Desaparecen los plugins y es reemplazado por un elemento llamado bundles.

Entre otras ventajas se destaca sobre todo la independencia de módulos que han creado. El propio núcleo del FrameWork está dividido en módulos que tienen una alta cohesión, lo que permite la reutilización de los mismos fuera de un proyecto basado en Symfony 2.

[6]**Potencial de uso:** Valor dado por el cálculo de la mediana. Representa el nivel obtenido según la cantidad de indicadores que posee.
[7]**Bundles**: Conjunto de archivos que implementan una única funcionalidad.

Además, el gran esfuerzo que han hecho en la seguridad del propio FrameWork – utilizando técnicas para evitar inyección SQL, XSS e incluso CSRF – así como la integración de PHP-Unit para la realización de pruebas unitarias.

Es un marco de trabajo muy documentado y se publica bajo licencia MIT22, con la que se puede desarrollar aplicaciones web comerciales gratuitas (22).

ORM Doctrine 2.

"Doctrine 2 es el mapeador de objetos relacionales (ORM) para PHP 5.3.0+ que provee una persistencia transparente para los objetos PHP. Este usa el patrón de mapeo de datos como el corazón de este proyecto, apuntando a una completa separación entre la lógica del dominio/negocio de la persistencia en un sistema de gestión de base de datos relacionales. El beneficio de Doctrine para los programadores es la habilidad de concentrarse únicamente en la lógica de negocio orientada a objetos y preocuparse de la persistencia solo como una tarea secundaria. Esto no significa que la persistencia no sea importante para Doctrine 2, sin embargo es nuestra creencia que existen considerables beneficios para la programación orientada a objetos si la persistencia y las entidades son perfectamente separadas." (23).

1.5.3. Lenguajes de programación.

Teniendo en cuanta que el Marco de trabajo seleccionado es un FrameWork PHP, se deduce que este sea el lenguaje a utilizar en el desarrollo de la solución.

Otros lenguajes utilizados.

CSS

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definirla presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).Permite separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. En este último podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style".

Las Ventajas de utilizar CSS

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Los Navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

HTML 5

HTML (HyperTextMarkupLanguage) en su versión 5, no sólo trata de incorporar nuevas etiquetas o eliminar otras, sino que supone mejoras en áreas que hasta ahora quedaban fuera del lenguaje y para las que se necesitaba utilizar otras tecnologías.

Los cambios comienzan añadiendo semántica y accesibilidad implícitas, especificando cada detalle y borrando cualquier ambigüedad. Se tiene en cuenta el dinamismo de muchos sitios webs, donde su aspecto y funcionalidad son más semejantes a aplicaciones webs que a documentos.

También cuenta con nuevas APIs para interfaz de usuario: temas tan utilizados como el "drag&drop" (arrastrar y soltar) en las interfaces de usuario de los programas convencionales, serán incorporadas al HTML 5 por medio de un API.

JavaScript

Se conoce como un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. JavaScript es interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Con JavaScript se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. Por su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de

práctica. Con JavaScript el programador, que se convierte en el verdadero dueño y controlador de cada cosa que ocurre en la página cuando la está visualizando el cliente. (24).

1.5.4. Servidores web.

Apache.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (HTTP) de la Apache Software Foundation. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración. Apache tiene amplia aceptación en la red: en el 2005, Apache fue el servidor HTTP más usado, siendo el servidor empleado en el 70% de los sitios web en el mundo. Sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden será provechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache. (25)

Ventajas

- Modular.
- Open source.
- Multi-plataforma.
- Extensible.
- Popular (fácil conseguir ayuda/soporte).

Lighttpd.

Lighttpd es un servidor web diseñado para ser rápido, seguro, flexible, y fiel a los estándares. Está optimizado para entornos donde la velocidad es muy importante, y por eso consume menos CPU y memoria RAM que otros servidores. Por todo lo que ofrece, lighttpd es apropiado para cualquier servidor que tenga problemas de carga. Según estadísticas en lighttpd.net, lighttpd es varias veces más rápido que Apache y thttpd en la mayoría de pruebas. (26)

Características

- Virtual hosting (alojar varios dominios en la misma IP).
- CGI, SCGI y FastCGI.
- Soporte para PHP, Ruby, y otros.
- Entorno chroot.
- Cifrado SSL.
- Compresión (gzip, bzip2,...).
- Autenticación (LDAP, htpasswd, otros).
- Server SideIncludes.
- Consumo de memoria constante.
- Redirecciones HTTP, y reescrituras de URL.
- Puede enviar partes de un fichero (rangos).
- También permite otro sistema de notificación de eventos como kqueue y epoll.
- Hace estadísticas mediante RRDtool.
- Muestra un listado de ficheros cuando se entra a un directorio sin index.html.
- Redirección condicional.
- Permite módulos externos.
- Acepta parte de WebDAV.

IIS: Internet Information Services.

Internet Information Server, IIS, es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS. Este servicio convierte a un ordenador en un servidor de Internet o Intranet decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web). El servidor web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas, por ejemplo Microsoft incluye los de Active Server Pages (ASP) yASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl. (27)

Thttpd.

Thttpd es un servidor web de código libre disponible para la mayoría de las variantes de Unix. Se caracteriza por ser simple, pequeño, portátil, rápido, y seguro, ya que utiliza los requerimientos mínimos de un servidor HTTP. Esto lo hace ideal para servir grandes volúmenes de información estática.

El uso apropiado para las personas que adoptan esta herramienta es de obtener velocidad en la transferencia de archivos y reducción de gastos innecesarios para funciones que no son requeridas en el servidor, debido a tener solo la posibilidad de utilizar servidores estándar (Apache).Este rasgo importante permite al administrador de servidor limitar la tarifa de bit máxima en la cual los ciertos tipos de archivos pueden ser transferidos, generando, una aplicación mucho más ligera y rápida. (28)

Características

- Simple, porque esto maneja sólo el mínimo necesario para poner en práctica el protocolo
- HTTP, algunas veces un poco más que el mínimo.
- Pequeño, porque esto también tiene un pequeño tamaño de período de explotación, ya que esto no se divide en dos partes y es muy cuidadoso sobre la asignación de memoria.
- Portátil, porque esto se compila limpiamente sobre la mayoría de sistemas operativos, expresamente incluyendo FreeBSD, SunOS 4, Solaris 2, BSD/OS, GNU/Linux, OSF.
- Rápido, porque en el empleo típico es sobre todo más rápido que los mejores servidores “destacados" (Apache), y bajo la carga extrema es mucho más rápido.
- Seguro, porque este se extiende a grandes longitudes para proteger el servidor Web contraataques de otros sitios.\

Ventajas

El administrador puede decidir restringir la transferencia de archivos de imagen JPEG a en la mayor parte de 20 kilobytes por segundo. Esto impide a la conexión hacerse saturado de modo que el servidor todavía sea sensible bajo la carga pesada, con la compensación que reducen la

velocidad de transferencia de archivo. Los promedios de carga se caen debido a la reducción de la transferencia grafica gracias a Thttpd.

Desventajas

No posee las mismas aplicaciones que se pueden obtener de un software estándar como lo es el Apache.

Servidor Web seleccionado (Apache)

Luego de hacerse un profundo análisis de los servidores Web: Apache, Lighttpd, IIS y Thttpd se ha decidido utilizar para el entorno a desarrollar el servidor Web Apache por las siguientes razones: Tiene una amplia aceptación en la red, siendo el servidor HTTP web más usado en los sitios web del mundo. La arquitectura de este servidor es muy modular y además presenta entre otras características mensajes de error altamente configurables. Las principales metas de su diseño son: velocidad, simplicidad, multiplataforma y facilidad de desarrollo distribuido.

El manejo y optimización de la cache del FrameWork seleccionado se integra fácilmente con este servidor web, permitiendo mejorar el rendimiento de la aplicación y el acceso a las rutas de la misma.

1.5.5. Gestor de base de datos.

MySQL

MySQL es un sistema de gestión de base de datos relacional, multi hilo y multiusuario con más de seis millones de instalaciones. MySQL AB desde enero de 2008 una subsidiaria de Sun Microsystems desarrolla MySQL como software libre en un esquema de licenciamiento dual. MySQL está escrito en una mezcla de C y C++. (29)

Las siguientes características son implementadas únicamente por MySQL:

- Múltiples motores de almacenamiento (MyISAM, Merge, InnoDB, BDB, Memory/heap, MySQLCluster, Federated, Archive, CSV, Blackhole y Example en 5.x), permitiendo al usuario escogerla que sea más adecuada para cada tabla de la base de datos.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD.

PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos. (30)

Características

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreignkeys).
- Disparadores (triggers): Un disparador o trigger se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

Ahora todos los disparadores se definen por seis características

- El nombre del trigger o disparador.
- El momento en que el disparador debe arrancar.
- El evento del disparador deberá activarse sobre...
- La tabla donde el disparador se activará.
- La frecuencia de la ejecución.
- La función que podría ser llamada.

Entonces combinando estas seis características, PostgreSQL le permitirá crear una amplia funcionalidad a través de su sistema de activación de disparadores (triggers).

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

Firebird

Firebird es un sistema de administración de base de datos relacional (o RDBMS) (Lenguaje consultas: SQL) de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en2000. Su código fue reescrito de C a C++ (31).

Características

- Es multiplataforma, y actualmente puede ejecutarse en los sistemas operativos: Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows.
- Ejecutable pequeño, con requerimientos de hardware bajos.
- Arquitectura Cliente/Servidor sobre protocolo TCP/IP.
- Soporte de transacciones ACID y claves foráneas.
- Es medianamente escalable.
- Buena seguridad basada en usuarios/roles.
- Diferentes arquitecturas, entre ellas el Firebird incrustado que permite ejecutar aplicaciones monousuario en ordenadores sin instalar el software Firebird.
- Bases de datos de sólo lectura, para aplicaciones que corran desde dispositivos sin capacidad de escritura, como cd-roms.
- Existencia de controladores ODBC, OLEDB, JDBC, PHP, Perl, .net, etc.
- Requisitos de administración bajos, siendo considerada como una base de datos libre de mantenimiento, al margen de la realización de copias de seguridad.
- Pleno soporte del estándar SQL-92, tanto de sintaxis como de tipos de datos.
- Completo lenguaje para la escritura de disparadores y procedimientos almacenados denominado PSQL.
- Capacidad de almacenar elementos BLOB (BinaryLargeOBjects).
- Soporte de User-Defined Functions (UDFs).
- Versión autoejecutable, sin instalación, excelente para la creación de catálogos en CD-ROM y para crear versiones de evaluación de algunas aplicaciones.

Servidor de Base de datos seleccionado (PostgreSQL)

Luego del estudio de varios servidores de bases de datos se ha seleccionado para la realización del sistema PostgreSQL debido a que:

- Es un sistema gestor de bases de datos “Open Source”.

- Presenta gran capacidad de almacenamiento, permitiendo al usuario escoger el que sea más adecuado para cada tabla de la base de datos.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.
- Alta ocurrencia y amplia variedad de tipos nativos.
- Además de las facilidades antes mencionadas se decidió escoger este gestor pues es muy usado en la UCI por ser muy rápido en aplicaciones de este tipo.

1.5.6. Metodologías.

RUP

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos (32).

MSF

La metodología MSF es del tipo de metodologías agiles, está enfocada a dirigir proyectos o soluciones de innovación, en ella no se detalla ni se hace énfasis de la organización ni el tamaño del equipo de desarrollo, está centrada en la gestión y administración del proyecto para lograr el impacto deseado. Involucra indudablemente la calidad ya que prevé liberar una solución si aún tiene fallos o desperfectos.

XP

La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Las cuatro variables de esta metodología son:

- **Coste**: Máquinas, especialistas y oficinas.
- **Tiempo**: Total y de Entregas.
- **Calidad**: Externa e Interna.
- **Alcance**: Intervención del cliente.

XP surgió como respuesta y posible solución a los problemas derivados del cambio en los requerimientos. Se plantea como una metodología a emplear en proyectos de riesgo. Con ella se aumenta la productividad (33).

Una Características básicasde XP es que la metodología se basa en (34):

- **Pruebas Unitarias**: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- **Re fabricación**: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **Programación en pares**: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Para facilitar la selección de la metodología se listan un conjunto de características que servirán para comparar las ventajas y desventajas de las metodologías RUP, XP y MSF, y se procede al cálculo de la mediana. Clasificación: **Alto**: 2, **Medio**: 1, **Bajo**: 0

Indicadores	**RUP**	**XP**	**MSF**
Desarrollo de aplicaciones web	Alto	Alto	Alto
Interacción del cliente con el equipo de desarrollo	Medio	Alto	Medio
Gestión de proyecto	Alto	Medio	Alto
Definición de requerimientos	Alto	Bajo	Alto
Facilidad para cambio de requerimientos	Medio	Alto	Medio
Facilidad para desarrollos de corta duración	Medio	Alto	Medio
Uso de herramientas libres	Alto	Alto	Bajo
Experiencia del equipo de desarrollo	Medio	Alto	Bajo
Mediana	**Alto**	**Alto**	**Medio**

Tabla 3 Análisis comparativo de las metodologías de desarrollo.

Para poder justificar la selección de la metodología en Tabla 4Tabla 3 se realiza nuevamente la comparación de las características descritas en la Tabla 3**¡Error! No se encuentra el origen**

de la referencia., tomando solo el valor cuantitativo que representa las clasificaciones dadas: **Bajo**: 0, **Medio**: 1, **Alto**: 2.

Indicadores	RUP	XP	MSF
Desarrollo de aplicaciones web	2	2	2
Interacción del cliente con el equipo de desarrollo	1	2	1
Gestión de proyecto	2	1	2
Definición de requerimientos	2	0	2
Facilidad para cambio de requerimientos	1	2	1
Facilidad para desarrollos de corta duración	1	2	1
Uso de herramientas libres	2	2	0
Experiencia del equipo de desarrollo	1	2	0
Puntaje	**12**	**13**	**9**

Tabla 4 Análisis comparativo de las metodologías de desarrollo con respecto al proyecto.

Como queda evidenciado, las metodologías RUP y XP presentan un Alto potencial de ser seleccionadas por cumplir con una mediana representativa de los indicadores definidos. De ambas XP obtiene mayor puntaje al realizar un análisis cuantitativo de la cantidad de indicadores de clasificación alta que posee.

Metodología seleccionada (XP)

Considerando la situación actual de la investigación y luego del estudio de las metodologías anteriores se decide utilizar XP ya que es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo y un pequeño equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

La metodología XP tiene un conjunto importante de reglas y prácticas. En forma genérica, se pueden agrupar en (35):

- Reglas y prácticas para la Planificación
- Reglas y prácticas para el Diseño
- Reglas y prácticas para el Desarrollo
- Reglas y prácticas para las Pruebas

Planificación

Se definen fechas de entrega de las primeras versiones del software, la organización del trabajo y el equipo de desarrollo y por último se detalla dentro de una versión del producto los problemas que deben de ser resueltos con carácter urgente (36).En esta fase se definen las historias de usuarios y se identifican el número y tamaño de las iteraciones

Historia de usuario

Las historias de los usuarios son descripciones que realiza el cliente en su propio lenguaje que da a conocer las necesidades del sistema. El nivel de detalle de las historias de usuario debe ser el mínimo posible, permitiendo hacerse una ligera idea de cuánto costará implementar el sistema (36).

Plan de iteraciones

Una vez identificadas las HU y estimado el esfuerzo propuesto para el desarrollo del sistema de cada una de estas historias, se procede a la planificación de la etapa de implementación del sistema. Teniendo en cuenta la prioridad que tiene una determinada HU en el desarrollo del componente se decide en que iteración será implementada. Las HU que cuentan con mayor importancia por ser funcionalidades indispensables para la aplicación deben ser implementadas en las primeras iteraciones del ciclo de desarrollo.

Diseño

El diseño adecuado para el software es aquel que supera con éxito todas las pruebas, refleja claramente la intención de implementación del equipo de desarrollo y tiene el menor número posible de clases y métodos. Unas de las partes importantes en XP es la simplicidad en todos los aspectos. Se considera que un diseño sencillo se logra más rápido y se implementa en menos tiempo. La complejidad innecesaria y el código extra debe ser removido inmediatamente (36).

Para el diseño de las aplicaciones, la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML, aunque el uso de estos diagramas puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante.

Tarjetas CRC (Cargo o Clase, Responsabilidad y Colaboración).

Las tarjetas CRC permiten desprenderse del método basado en procedimientos y trabajar con una metodología basada en objetos, además de posibilitar que el equipo completo contribuya en la tarea del diseño. Las tarjetas CRC representan objetos; se utilizan normalmente cuando se determinan las clases necesarias y cómo van a interactuar, luego se implementa la solución.

Implementación

La implementación o codificación es el flujo de trabajo donde se producen los componentes del sistema: ejecutables, ficheros de código fuente, scripts, entre otros. Tiene como objetivo principal desarrollar la arquitectura y el sistema como un todo, así como definir la organización del código. La Metodología XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para incrementar la visión de los desarrolladores con la opinión de éste (37).

1.6. Herramientas utilizadas.

Una herramienta es un dispositivo o procedimiento que aumenta la capacidad de llevar a cabo determinadas tareas, por ejemplo herramientas de programación, de gestión, matemáticas, entre otras (38).

NetBeans

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso (39).

Entre las novedades principales de esta versión destaca mejoras en el soporte para la nueva sintaxis de Java 7, soporte de HTML5, mejoras al editor Java, mejoras en la integración con WebLogic y soporte para PHP 5.4. Otras características que presenta son:

- Tienen soporte para las Base de datos de Oracle.
- El equipo de desarrollo del soporte para PHP de NetBeans 7.* incluye una interesante característica que permite definir tipos de variables en lo comentarios. Esto es útil, por ejemplo, en aquellas situaciones en los que el código de terminación no reconoce el tipo de objeto que se está tratando.
- Tiene una sección de edición de HTML5.

- Netbeans 7.3 está disponible para usuarios de sistemas operativos Windows, Mac, Linux y Solaris (39).

Subversión

Para el control de versiones según las necesidades del proyecto se usa Subversión, es un software libre bajo una licencia de tipo Apache/BSD y se le conoce también como SVN por ser el nombre de la herramienta utilizada en la línea de comando.

Entre algunas de las ventajas por las que se selecciona esta herramienta están:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.).

1.7. Arquitectura del sistema.

La arquitectura permite descomponer el sistema en piezas que agrupan aspectos específicos del mismo, producto de un proceso de abstracción y que al organizarse de cierta manera constituyen la base de la solución de un problema en particular.

Se adopta el modelo “4+1” de Kruchten, este es un modelo de vistas (40) diseñado por el profesor Philippe Kruchten y que encaja con el estándar “IEEE 1471-2000” (41) que se utiliza para describir la arquitectura de un sistema software intensivo basado en el uso de múltiples puntos de vista. Estas 4 vista las denominó Kruchten como: **vista lógica**, **vista de procesos**, **vista de despliegue** y **vista física** y la vista “+1”que tiene la función de relacionar las 4 vistas anteriores, se denomina **vista de escenario**.

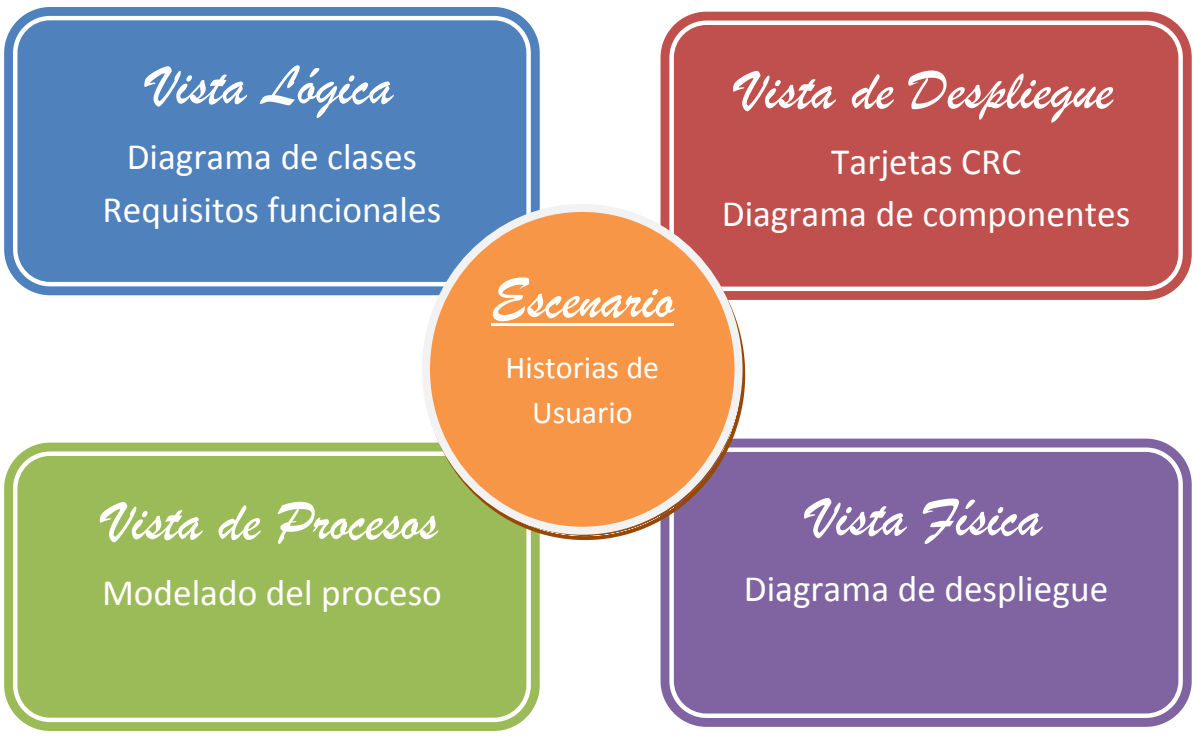


Figura 2 Modelo 4+1 vista. Representación de la arquitectura del sistema.

1.8. Pruebas.

Las pruebas son un paso importante en el proceso de culminación de un sistema informático ya que permiten aumentar la calidad de los mismos reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

La metodología XP divide las pruebas del sistema en dos grupos las pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final (36).

1.8.1. Pruebas funcionales o de aceptación.

Las pruebas funcionales o de aceptación son la mejor forma de probar que se tiene un sistema informático con las características que desea el cliente, desde la petición realizada por el navegador hasta la respuesta enviada por el servidor. Las pruebas funcionales prueban todas

las capas del sistema informático dígase el sistema de enrutamiento, el modelo, las acciones y las plantillas (36).

Las pruebas de aceptación son una parte integral del desarrollo incremental según lo practicado por la metodología XP. Todas las historias de usuario son apoyadas por las pruebas de aceptación, las cuales son definidas por el cliente en el sistema. Estas pruebas se realizan frente a los temores de que el negocio ha sido mal entendido por parte del equipo de desarrollo. Las pruebas de aceptación significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente. Las pruebas de aceptación exigen al cliente a profundizar en su conocimiento del dominio y, precisamente, afirmar lo que el sistema informático debe hacer en circunstancias específicas, por esto, el cliente es la persona adecuada para diseñar las pruebas de aceptación.

1.8.2. Pruebas unitarias.

Las pruebas unitarias son las formas de probar el correcto funcionamiento en código de un sistema informático, facilitando que cada uno de los módulos que componen dicho sistema funcione correctamente por separado.

En la presente investigación se propone ir añadiendo pruebas unitarias para que se encuentren y solucionen errores en el sistema informático propuesto trayendo consigo que después de distintas iteraciones el código no sólo sea adecuado, sino que cada vez sea mayor el porcentaje del sistema que este cubierto por pruebas.

La solución informática incluirá estas pruebas haciendo uso de las siguientes métricas: **Tamaño operacional de la clase (TOC):** Está dada por el número de funcionalidades asignadas a cada una de las clases del sistema informático propuesto en la investigación evaluando los siguientes atributos de calidad:

- **responsabilidad:** Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **complejidad de implementación:** Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización**: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

1.9. Conclusiones.

En este capítulo quedan descritos los elementos necesarios para comprender la actual situación y determinar cuál o cuáles son las necesidades a satisfacer. Partiendo del análisis comparativo, con el uso de indicadores, aplicado a varios software existentes se define una propuesta de solución basada en la necesidad de llevar a cabo la implementación de una aplicación web utilizando algunas características de SiGEst. El análisis y comparación de varias herramientas de trabajo arrojan como resultado la selección de un entorno de desarrollo altamente integrado y compatible entre sí.

CAPÍTULO 2: PLANIFICACIÓN, DISEÑO E IMPLEMENTACIÓN.

2.2. Introducción.

Durante este capítulo se obtendrán un conjunto de artefactos de la metodología XP que serán generados en cada fase y ayudaran en el desarrollo del sistema informático. En la fase de Planificación se describen elementos como Historias de Usuarios, Plan de iteraciones y plan de entregas. En la fase de diseño se definen las metáforas del sistema y las tarjetas CRC y en la fase de desarrollo estaremos viendo las tareas definidas para cada historia de usuario.

Fases y artefactos generados en la metodología XP

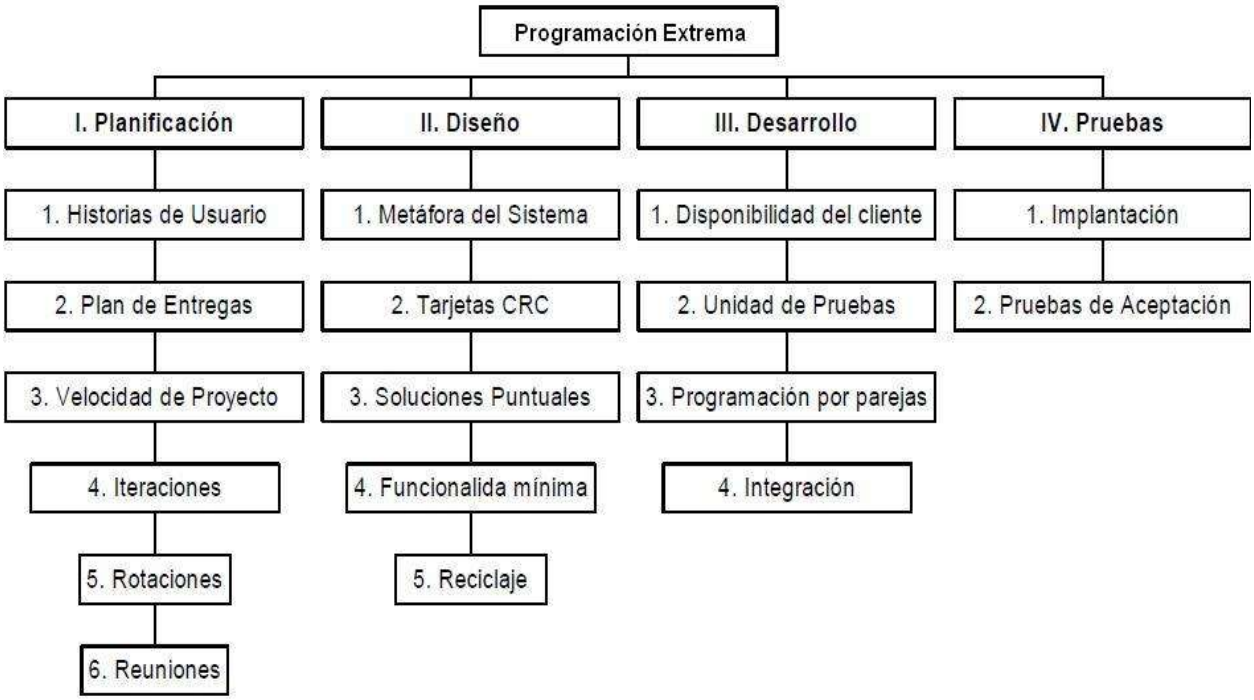


Figura 3 Fases de la metodología XP

2.3. Planificación

Durante este epígrafe se mostrará lo que sucede en la fase de planeación de forma práctica contrastándola con la descripción teórica que se hace de esta fase en el Capítulo 1, el epígrafe 1.5.6 En esta fase fue necesaria la constante interacción con el cliente, basado en esto se logró

recolectar las historias de usuarios rápidamente y se determinó el número y tamaño de las iteraciones y se determinó las fechas de las primeras entregas.

2.3.1. Historias de usuarios.

El cliente realizó una descripción de las historias de usuarios (HU) con el mínimo nivel de detalle ayudado por el equipo de desarrollo teniendo en cuenta las funciones que debía realizar para lograr identificarlas. También es necesario destacar que las HU han jugado un papel importante en la estimación de los tiempos requeridos para el desarrollo del proyecto.

A continuación se muestran dos ejemplos de los contenidos de las fichas de las HU, para consultar el resto debe remitirse al Anexo

Anexo 1 Fichas de historias de usuarios., al final del documento. Estas fichas quedan estructuradas de la siguiente forma:

Número: A cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.

Usuario: Es el usuario del sistema que define la HU.

Nombre: Nombre descriptivo de la HU.

Programador: Nombre del programador encargado de desarrollar la HU.

Prioridad en Negocio: Grado de prioridad que le asigna el cliente a la HU en dependencia del valor en el negocio. Los valores que puede tomar son (Alta, Media o Baja).

Riesgo en Desarrollo: Grado de complejidad que le asigna el equipo de desarrollo a la HU luego de analizarla. (Alto, Medio o Bajo).

Puntos Estimados: Esfuerzo estimado por el equipo de desarrollo para darle cumplimiento a la HU.

Iteración Asignada: Número de la iteración en la cual será implementada la HU.

Descripción: Descripción simple sobre lo que debe hacer la funcionalidad a la que se hace referencia.

Historia de Usuario	
Número: 2	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Planificar guardia a estudiantes	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 8
Riesgo en Desarrollo: Alto **(Alto/Medio/Bajo)**	**Iteración Asignada:**1
Descripción: Inicia cuando el Administrador de área decide planificar la guardia a los estudiantes en su área. Para esto necesita el listado de estudiantes, de postas y turnos con los que dispone y definir un rango de tiempo a planificar.	
Observaciones:	

Tabla 5 Historia de usuario Planificar guardia a estudiantes.

Historia de Usuario	
Número: 7	**Usuario:** Controlador.
Nombre de Historia de Usuario: Dar evaluación	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada:**3
Descripción: Inicia cuando el controlador decide evaluar una guardia. Para esto necesita tener la lista de las guardias que controla y no ha evaluado aún.	
Observaciones:	

Tabla 6 Historia de Usuario Evaluar guardia.

Luego de ser recolectadas las HU el equipo de desarrollo se reúne para estimar cuanto tiempo, idealmente, llevaría implementar cada una. El proceso de estimación se realizó utilizando la técnica Planning Poker o póker de planificación.

El objetivo del Planning Poker es obtener una medida de tamaño relativo de todas las HU respecto a sí mismas mediante la utilización de puntos, esta técnica es explicada en la

fundamentación teórica del capítulo 1 el epígrafe 1.5.6 Cada punto significa un peso, un esfuerzo o complejidad para completar un objetivo. La numeración está basada en la sucesión de Fibonacci.[8] La distancia entre números crece conforme se hacen mayores, de esta manera, se facilita la decisión sobre qué tamaño tiene un objetivo.

También fue necesaria la priorización de las HU, esto se hizo teniendo en cuenta el valor en el negocio de cada historia para el cliente. Al finalizar se obtuvo una lista de la HU donde se refleja los siguientes campos:

Número: Es el número asignado a la HU para identificarla

Nombre: Nombre de la HU que ha sido definido teniendo en cuenta la función que debía realizar.

Prioridad: Valor en el negocio que tiene la HU para el cliente.

Esfuerzo: Valor estimado mediante puntos de HU.

Número	Nombre	Prioridad	Esfuerzo
1	Administración de áreas	Alta	3
2	Administración de configuraciones del área	Alta	3
3	Planificar guardia a estudiantes	Alta	8
4	Planificar guardia a trabajadores	Alta	8
5	Planificar guardia a controladores	Alta	6
6	Administrar planificación	Alta	2
7	Evaluar guardia	Alta	2
8	Emitir evaluación	Alta	2
9	Reportar incidencias	Alta	2
10	Administrar incidencia	Alta	3
11	Reportar a comisión disciplinaria y demerito.	Media	3
12	Solicitud y cabio de guardia	Media	3
13	Administración de cambios de guardia	Media	2

[8]En matemáticas, la sucesión de Fibonacci es la sucesión infinita de números naturales: 0,1,1,2,3,5,8,13,21,34,55,89,144… La sucesión comienza con los números 0 y 1, y a partir de estos, cada elemento es la suma de los dos anteriores.

14	Visualizar mis evaluaciones	Baja	2
15	Ajuste de planificación	Baja	3
16	Emitir reportes	Baja	2
17	Visualizar estadísticas	Baja	2

Tabla 7 Historias de usuarios detectadas.

Se obtuvo un total de 17 HU con un tiempo estimado para el desarrollo del sistema de 56 puntos.

2.3.2. Requisitos Funcionales

Una vez realizadas por el cliente las distintas historias de usuarios se procede a documentar las necesidades del mismo capturando los requisitos identificados a partir de las historias de usuario. Los requisitos son la base para un desarrollo exitoso así como para una plena conformidad con el entregable final. A continuación se muestra el listado de requisitos definidos para el sistema informático propuesto por la investigación.

RF _Autenticar usuario

RF_Crear nuevo usuario.

RF_Eliminar usuario.

RF_Crear una nueva posta.

RF_Eliminar una posta.

RF_Crear un nuevo turno.

RF_Eliminar un turno.

RF_Crear una nueva área.

RF_Eliminar un área.

RF_Planificar guardia a estudiantes.

RF_Planificar guardia a trabajadores.

RF_Planificar guardia a controladores.

RF__Publicar una planificación.

RF_Eliminar una planificación.

RF_Modificar planificación.

RF_Evaluar guardia.

RF_Modificaruna evaluación.

RF_Crear una incidencia.

RF_Crear respuesta a una incidencia.

RF_Eliminar una incidencia.

RF_Modificar respuesta a una incidencia.

RF_Reportar a comisión disciplinaria.

RF_Eliminar reporte a comisión.	RF_ Aceptar solicitud.
RF_Reportar a demerito.	RF_Listar evaluaciones.
RF_Eliminar reporte a demerito.	RF_Generar reportes.
RF_Solicitar cambio de guardia.	RF_Generar estadísticas.

2.3.3. Requisitos no funcionales del sistema.

Requerimientos de apariencia o interfaz externa

- La interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo con la misma.
- La comunicación entre el servidor Web y las máquinas clientes será mediante el HTTP y entre el servidor y el directorio activo mediante LDAP.
- Las interfaces tienen que ostentar un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad de tal manera que no se haga difícil para los usuarios la utilización del sistema.

Requerimientos de usabilidad

- La aplicación propuesta será usada por personas que tengan o no conocimientos básicos de informática.
- El sistema estará disponible las 24 h del día.
- A los administradores del sistema se les dará un adiestramiento básico en el uso de la aplicación.
- Tendrán un nivel de acceso amplio en la aplicación.
- A los usuarios del sistema se les proveerá de una guía del sistema.

Requerimientos de rendimiento

- Para un buen funcionamiento de la aplicación se seguirán las diferentes técnicas de elaboración de sitios web, que faciliten el acceso rápido a sus páginas.
- La eficiencia del producto estará determinada en gran medida por el aprovechamiento de los recursos y la velocidad de la consultas de la base de datos.

- La aplicación propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible (menor a los 30seg.), adecuado a la rapidez con que el cliente requiere la respuesta a su petición.

Requerimientos de soporte

- Se necesita un servidor de bases de datos que soporte volúmenes de datos (PostgreSQL 9.*).
- Se necesita un servidor web (Apache 2.*)
- El sistema será multiplataforma (Linux o Windows).

Requerimientos de seguridad

- La información manejada por el sistema debe estar protegida de acceso no autorizado.
- La aplicación deberá estar disponible en todo momento para aquellas personas con acceso a la información.
- La seguridad se establecerá por roles que se le asignarán a los usuarios que interactúen con el sistema.
- El software brindará solamente aquellas funcionalidades que competen a la unidad ejecutora donde esté implantado.

Requerimientos de software

- Un servidor web con los módulos: php5-xsl, php5-pgsql, php5-mencache, php5-cli, php-apc, php-soap y el php5-intl.
- Un servidor de Base de dato y copia de seguridad.
- Pc clientes con un navegador web (Firefox 20.0 o superior, Chrome 20 o superior).

Requerimientos de hardware

- En el cliente se requiere una máquina con 256 MB de RAM como mínimo.
- Procesador Intel® a 1 GHz. de velocidad de procesamiento o superior.
- Tarjeta de red Ethernet.
- Todas las máquinas implicadas en la funcionalidad de la aplicación deben estar conectadas a la red.

- El servidor requiere mínimo 256 MB de RAM, procesador a 3 GHz de velocidad y tarjeta de red Ethernet.

2.3.4. Plan de iteración.

Una vez determinada la prioridad de las HU, los clientes deciden cuales realizar primero y se planifica su desarrollo en la primera iteración, luego de esto se determina la velocidad del proyectos, que no es más que la suma del esfuerzo de las HU, completadas al final de la primera iteración y se utiliza para planificar las iteraciones siguientes.

Se determinó desarrollar las HU administración de área y gestión de la configuración del área en la primera iteración por tratarse de actividades vitales y que dan continuidad al resto. Además también se desarrolla en esta iteración la HU planificar guardia a estudiantes por ser una de las más importantes del negocio, en las demás iteraciones se distribuyeron en dependencia de lo que ya se había implementado. Cuando tomamos el total de puntos estimados entre la velocidad del proyecto determinada en la primera iteración, definimos cuantas iteraciones son necesarias para el desarrollo de las HU. También se usará la velocidad del proyecto para determinar si una iteración está sobrecargada. La suma de los días que costará desarrollar todas las tareas de la iteración no debería sobrepasar la velocidad del proyecto de la iteración anterior.

Luego de realizar el cálculo anteriormente descrito el equipo de desarrollo precede a dividir la confección del sistema informático en 4 iteraciones. A continuación se muestra como queda conformado el plan de iteraciones que tiene los siguientes campos.

Iteración: Iteración que se define.

Historia de usuario: Nombre de la HU.

Prioridad: Valor en el negocio que tiene la HU para el cliente.

Esfuerzo: Valor estimado mediante puntos de HU.

Tiempo de duración: Suma del esfuerzo de las HU en cada iteración.

Iteración	Historia de usuario	Prioridad	Esfuerzo	Tiempo de duración
Iteración 1	Administración de áreas	Alta	3	14
	Administración de configuraciones del área	Alta	3	
	Planificar guardia a estudiantes	Alta	8	
Iteración 2	Planificar guardia a trabajadores	Alta	8	14
	Planificar guardia a controladores	Alta	6	
Iteración 3	Administrar planificación	Alta	2	14
	Evaluar guardia	Alta	2	
	Emitir evaluación	Alta	2	
	Reportar incidencias	Alta	2	
	Administrar incidencia	Alta	3	
	Reportar a comisión disciplinaria y demerito.	Media	3	
Iteración 4	Solicitud y cabio de guardia	Media	3	14
	Administración de cambios de guardia	Media	2	
	Visualizar mis evaluaciones	Baja	2	
	Ajuste de planificación	Baja	3	
	Emitir reportes	Baja	2	
	Visualizar estadísticas	Baja	2	

Tabla 8 Plan de iteraciones

2.3.5. Plan de entrega.

Ya elaboradas por el cliente las distintas HU y confeccionado por los desarrolladores el plan de iteración se procede a la confección del plan de entrega. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores, fijándose un periodo de tiempo determinado por puntos sobre el cual se debe implementar cada HU. El equipo de desarrollo determinó que cada punto equivale a un día ideal de trabajo.

Para aproximar el tiempo de la iteración se tomó como medida una semana, cada semana consta de cinco días en los que se trabajaba 8 horas sin distracciones, esta decisión fue acogida por el equipo debido a las obligaciones externas al proyecto.

A continuación veremos cómo queda elaborado el plan de entrega que tiene los siguientes campos:

No.: Numero de la iteración.

Historia de usuario: Nombre de la HU.

Prioridad: Valor en el negocio que tiene la HU para el cliente.

Esfuerzo: Valor estimado mediante puntos de HU.

Duración (**semanas**): Tiempo aproximado de la duración de la iteración en semanas.

Fecha de inicio: Fecha en la que comienza a desarrollarse la iteración.

Fecha de Fin: Fecha de entrega de la iteración.

No.	Historia de usuario	Esfuerzo	duración	Fecha inicio	Fecha Fin
1	Administración de áreas	3	2.4	18/03/2013	04/04/2013
	Administración de configuraciones del área	3			
	Planificar guardia a estudiantes	8			
2	Planificar guardia a trabajadores	8	2.4	08/04/2013	25/04/2013
	Planificar guardia a controladores	6			
3	Administrar planificación	2	2.4	19/04/2013	16/05/2013
	Evaluar guardia	2			
	Dar evaluación	2			
	Reportar incidencias	2			
	Administrar incidencia	3			
	Reportar a comisión disciplinaria y demerito.	3			

4	Solicitud y cabio de guardia	3	2.4	20/05/2013	06/06/2013
	Administración de cambios de guardia	2			
	Visualizar mis evaluaciones	2			
	Ajuste de planificación	3			
	Emitir reportes	2			
	Visualizar estadísticas	2			
	Solicitud y cabio de guardia	3			

Tabla 9 Plan de entrega

2.4. Diseño

El diseño es realizado durante todo el tiempo de vida del proyecto, siendo revisado y modificado debido a cambios presentados durante el desarrollo de forma casi constante. Este epígrafe muestra los resultados prácticos de la descripción teórica sobre el diseño usándola metodología XP que se hace en el Capítulo 1 el epígrafe: 1.5.6Metodologías.

Durante el diseño se invirtió el tiempo exclusivamente necesario en la elaboración de diagramas y diseños de interfaces gráficas. No se invertido mucho tiempo en el diseño de interfaces, pero se trató de ubicar los elementos según el cliente los solicitó. En los que refiere a diagramas se crearon tarjetas CRC y el modelo entidad-relación del cual han surgido varias versiones a la medida que se han incorporado funcionalidades a la aplicación.

2.4.1. Tarjetas CRC

Las tarjetas CRC fuero las bases para la obtención del modelo entidad relación. Cada tarjeta se convirtió en objeto sus responsabilidades en métodos públicos y sus colaboradores en llamados a otras clases. Durante la elaboración de las tarjetas los dos miembros del equipo estuvieron presentes y esto favoreció un mejor entendimiento.

En XP el proceso de diseño es iterativo por lo que la creación de las tarjetas no es en un mismo tiempo, se crean según las iteraciones y se le añaden responsabilidades y colaboradores según se haga necesario. A continuación se muestran las clases de diseño identificadas por el equipo de desarrollo así como dos ejemplos de las tarjetas CRC, el resto podrá verse en el anexo 2 al final del documento.

Clases:

- ✓ Área.
- ✓ Posta.
- ✓ Incidencias.
- ✓ Guardia.
- ✓ SolicitudDeCambio
- ✓ Evaluación
- ✓ Usuario
- ✓ Turno
- ✓ Planificación

En las Tarjetas CRC el nombre de la clase se coloca a modo de título, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha.

Clase: Nombre de la clase con que se está modelando.

Responsabilidades: Las responsabilidades de una clase son las cosas que conoce y las que realizan, sus atributos y métodos.

Colaboradores: los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

Clase Planificación	
Responsabilidad	**Colaboración**
show () new () create () edit() update() delete() turnosXDias() cantUsuariosNecesarios() cantDiasPlanificar() distribControladores()	 Usuario Turno Posta Evaluación Guardia

Tabla 10 Tarjeta CRC: Planificación

2.4.2. Diagrama de clases

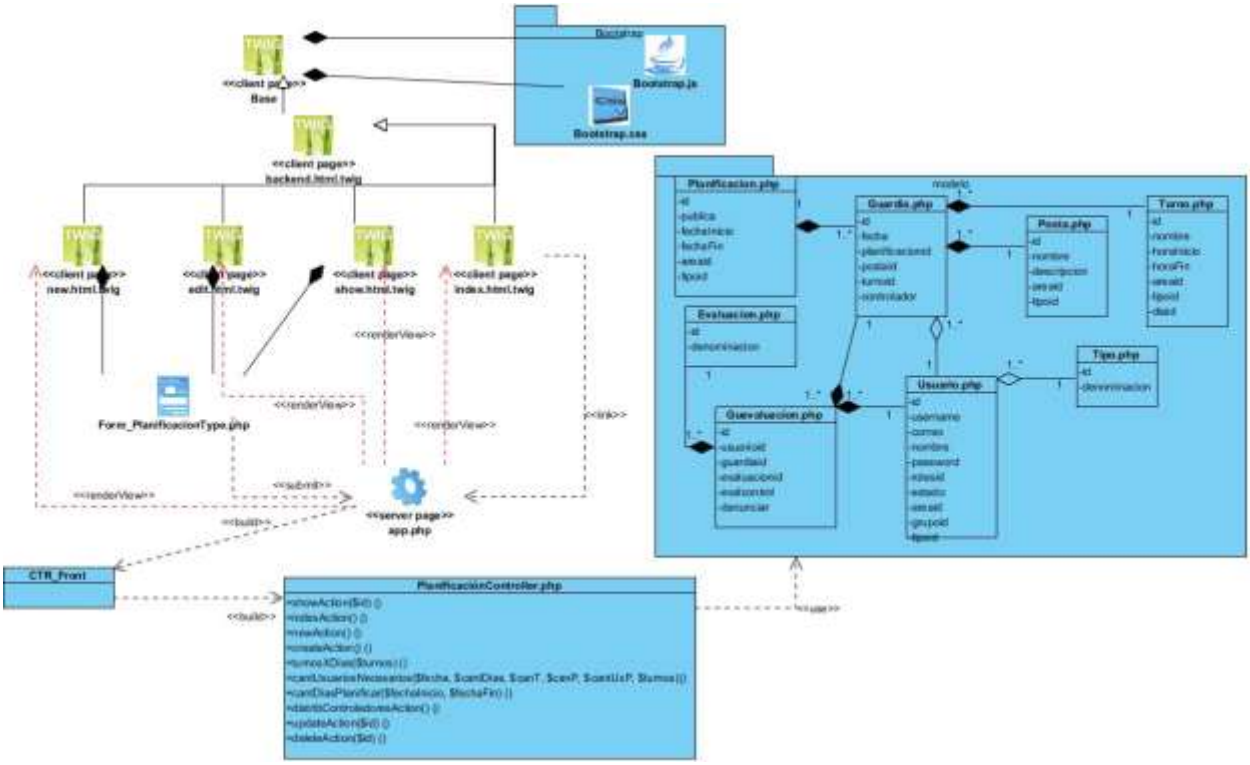


Figura 4 Diagrama de clases con estereotipos web para la administración de planificación

Se obtuvo un diagrama de clases con estereotipos web donde se representan las relaciones entre las clases del proceso de planificación en el sistema como parte de la vista lógica del modelo arquitectónico utilizado.

2.4.3. Modelo de datos

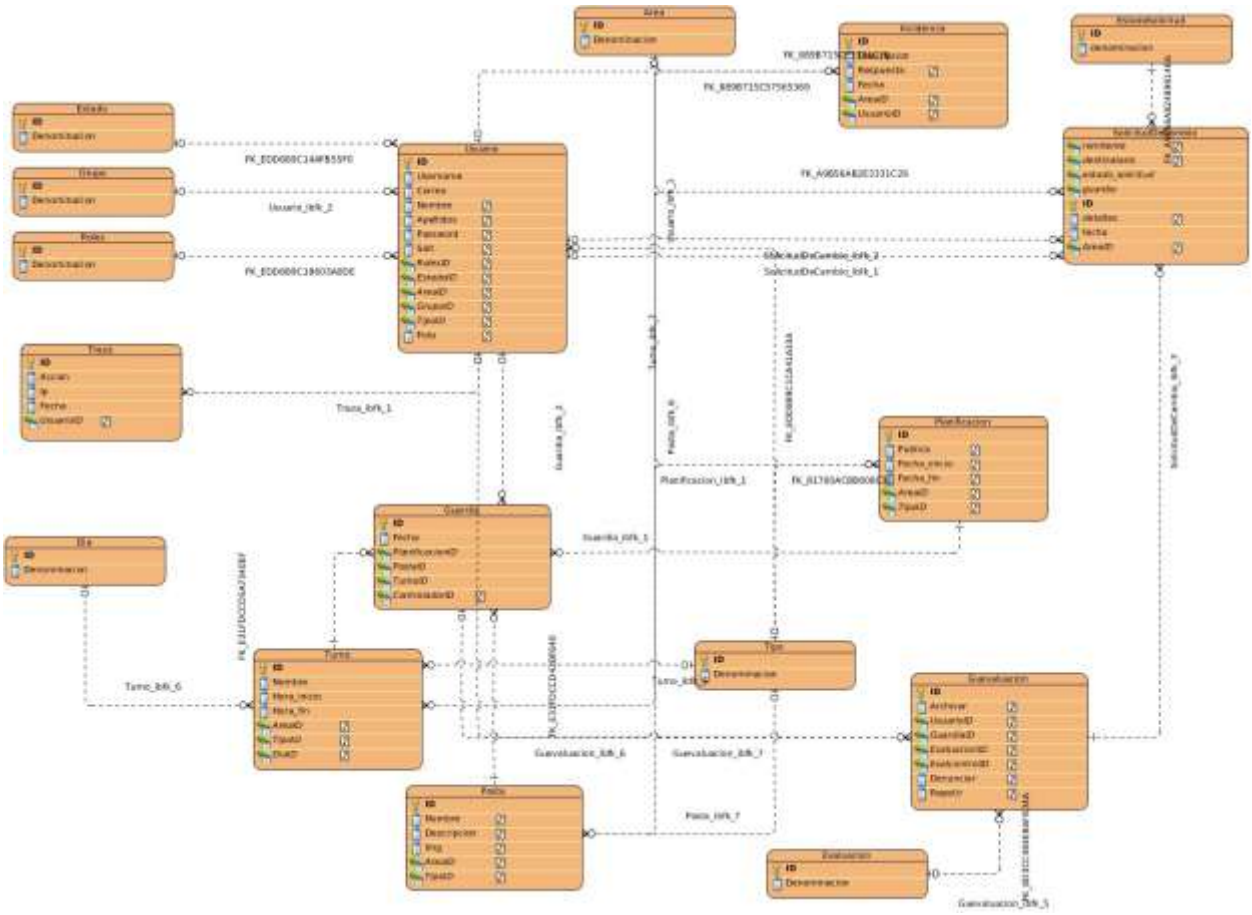


Figura 5 Modelo de datos

Este modelo de datos representa las tablas (entidades) importantes para el funcionamiento del negocio y muestra los datos que serán contenidos en el sistema. El modelo de datos obtenido cuenta con un total de 17 tablas donde 8 de ellas son nomencladores encargados de gestionar conceptos específicos del negocio, ya predefinidos como ejemplo tenemos la tabla Tipo, que se relaciona con la tabla usuario y se definen los tipos de usuario que existirán en el sistema (estudiantes, trabajadores). Las restantes tablas facilitan el registro de datos que son necesarios para el funcionamiento de la aplicación, por ejemplo la tabla planificación registra datos como la fecha de inicio, fecha de fin, área a la que pertenece y si es pública o no. La tabla guardia es la encargada de almacenar todas las guardias que se planifiquen.

2.4.4. Patrones de diseño.

Patrón de Arquitectura

Los patrones arquitectónicos son aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software. Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades, relaciones, y las formas en que colaboran (42).

Patrón Modelo-Vista-Controlador (MVC)

Patrón MVC (Modelo-Vista-Controlador) se encarga de separar interfaces, fue creado con Smalltalk-80[9]. El modelo es la capa del dominio, la vista es la capa de presentación y el controlador son los objetos de flujo de trabajo (42).

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Teniendo en cuenta que es una aplicación web la que se desarrolla, se ve el uso de este patrón de llamada y retorno MVC (según CMU[10]), donde el modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos, la vista transforma esta información obtenida por el modelo en las páginas web a las que acceden los usuarios y el controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista.

Patrones de diseño que implementa Symfony

El marco de trabajo Symfony utiliza varios patrones de diseño a continuación se ejemplificara cuales son y cómo son utilizados

Inyección de Dependencias: Permite desacoplar unos componentes de otros y reducir la interdependencia entre objetos y librerías. Consiste en pasar a cada componente todo lo que necesita a través de sus constructores, métodos o campos. Se deben diseñar los servicios de

[9] Smalltalk-80: Lenguaje de programación reflexivo orientado a objetos, de tipo dinámico.

[10] CMU: Universidad Carnegie Mellon (en inglés: Carnegie Mellon University, CMU), ubicada en la ciudad de Pittsburgh (Pensilvania) y es uno de los más destacados centros de investigación superior de los Estados Unidos en el área de informática y robótica.

manera que no construyan ellos mismos los servicios de los cuales dependen, sino que los servicios dependientes se pasen debidamente construidos. Permite estandarizar y centralizar la forma en que se construyen los objetos en la aplicación. (43).Este patrón se utilizó en la creación de servicios en la arquitectura base que serán utilizados por todos los módulos del proyecto.

Unidad de trabajo: Ya que Doctrine es consciente de todas las entidades gestionadas, cuando se llama al método flush(), calcula el conjunto de cambios y ejecuta la(s) consulta(s) más eficiente(s) posible(s). Por ejemplo, si se persiste un total de 100 objetos y, posteriormente se llama a flush(), Doctrine creará una sola declaración preparada y la volverá a utilizar en cada inserción. Este patrón se conoce como Unidad de trabajo, y se utiliza porque es rápido y eficiente (44).

Patrones GRASP utilizados:

Experto: Es uno de los más utilizados, puesto que Symfony utiliza el ORM Doctrine para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades. Este patrón se pone de manifiesto en la implementación de las clases persistentes como por ejemplo la clase planificación para ver una planificación es necesario listar todas las instancias de guardias asociadas a esa planificación, una instancia de planificación contiene toda la información necesaria para esto, por lo que el objeto planificación es adecuado para esta responsabilidad; es un experto en información para el trabajo. En este ejemplo podemos ver también que para listar las guardias asociadas a la planificación es necesario conocer las postas y los turnos, por lo que una instancia de guardia contiene toda la información necesaria, y esto la convierte en adecuada para esta responsabilidad por tanto en experta en esta información.

En las clases controladoras se encuentran las acciones definidas para el sistema y que se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, tenemos como ejemplo la clase PlanificadorController donde podemos evidenciar que es “creador” de dichas entidades. Permite especificar los argumentos pasados al constructor, así como llamar a los métodos y establecer parámetros.

Alta Cohesión: Una de las características principales de Symfony es la organización del trabajo en él mismo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, las clases con sufijo Controller en su nombre contienen varias funcionalidades estrechamente relacionadas entre ellas, teniendo un sentido común y un propósito único, siendo las encargadas de controlar las acciones de las plantillas y por lo tanto pertenecen a la capa del Controlador dentro de la arquitectura Modelo-Vista-Controlador. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.

Bajo Acoplamiento: Las clases*Controller* heredan solamente de *BaseController* para lograr un bajo acoplamiento de clases. También en la entidad Usuario se pone de manifiesto al implementar la clase UserInterface de Symfony. Ver Figura 11Ejemplo del patrón GRASP Bajo Acoplamiento en el anexo 3.

Controlador: Todas las peticiones Web son manejadas por un solo controlador frontal (app.php), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario. Ver Figura 6 Diagrama de componentes.

Patrones GOF utilizados:

En la categoría Comportamiento:

Observador: Cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Un complemento debe ser capaz de agregar métodos, o hacer algo antes o después de ejecutar un método, sin interferir con otros complementos. Tomemos como ejemplo la clase TrazaListenque es la encargada de registrar todos los eventos ocurridos en las demás clases.

Decorador: Este método pertenece a la clase abstracta sfView, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado layout.php es el que contiene el Layout de la página. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla.

En la categoría Creacionales:

Abstract Factory (fábrica abstracta): su propósito es definir una interfaz para la creación de familias de objetos relacionados o dependientes sin tener que especificar la clase concreta. La fábrica abstracta se utiliza para la creación de los objetos entidades. También se utiliza en las clases controladoras para la creación de formularios, haciendo uso del método createform(), pasándole el formulario a crear y el objeto de la entidad correspondiente.

2.5. Implementación

Algunos de los elementos en cuanto a implementación más importantes son, que el cliente debe estar presente todo el tiempo, se debe trabajar en parejas y debe haber una propiedad colectiva del código para su mejor entendimiento. A continuación se verá como fueron aplicadas las ideas teóricas expuestas en el Capítulo1 epígrafe 1.5.6 sobre implementación.

En esta fase se genera todo el código fuente necesario para satisfacer las HU definidas para la solución y se describen todas las tareas realizadas en cada iteración. Al inicio de cada iteración, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario. Todas las HU son traducidas en tareas de programación, cada una de ellas se asignó a un programador.

2.5.1. Diagrama de componente

Un diagrama de componentes representa la separación de un sistema de software en componentes físicos (por ejemplo archivos, módulos, paquetes, etc.) y muestra las dependencias y organización entre estos componentes.

El diagrama de componentes del sistema de GOE evidencia las interacciones entre el los componentes de Symfony y el módulo GOE. Symfony cuenta con los componentes *Security* (para la seguridad), *Routing* (para las rutas de acceso), *Config* (para las configuraciones), *Mensajería* (para la gestión de mensajes informativos y de errores), *Excepciones* (para el manejo de las excepciones generadas) y *Ctr_Frontal* es el controlador frontal que recibe cada petición realizada al subsistema, entre otros. Estos componentes permiten un adecuado manejo de la seguridad, las configuraciones, las excepciones y errores del subsistema.

El módulo GOE está compuesto por cuatro paquetes fundamentales, FredBandel (contiene el componentes LDAP para la conexión al directorio activo de la UCI) Controlador (encargado de dar respuesta a las peticiones de los usuarios), Modelo (contiene las clases entidades del modelo), Vista (encargado de la construcción y manejo de las interfaces de usuario). También existe una relación del paquete Controlador con el componente Routing para el manejo de las rutas definidas para el modulo.

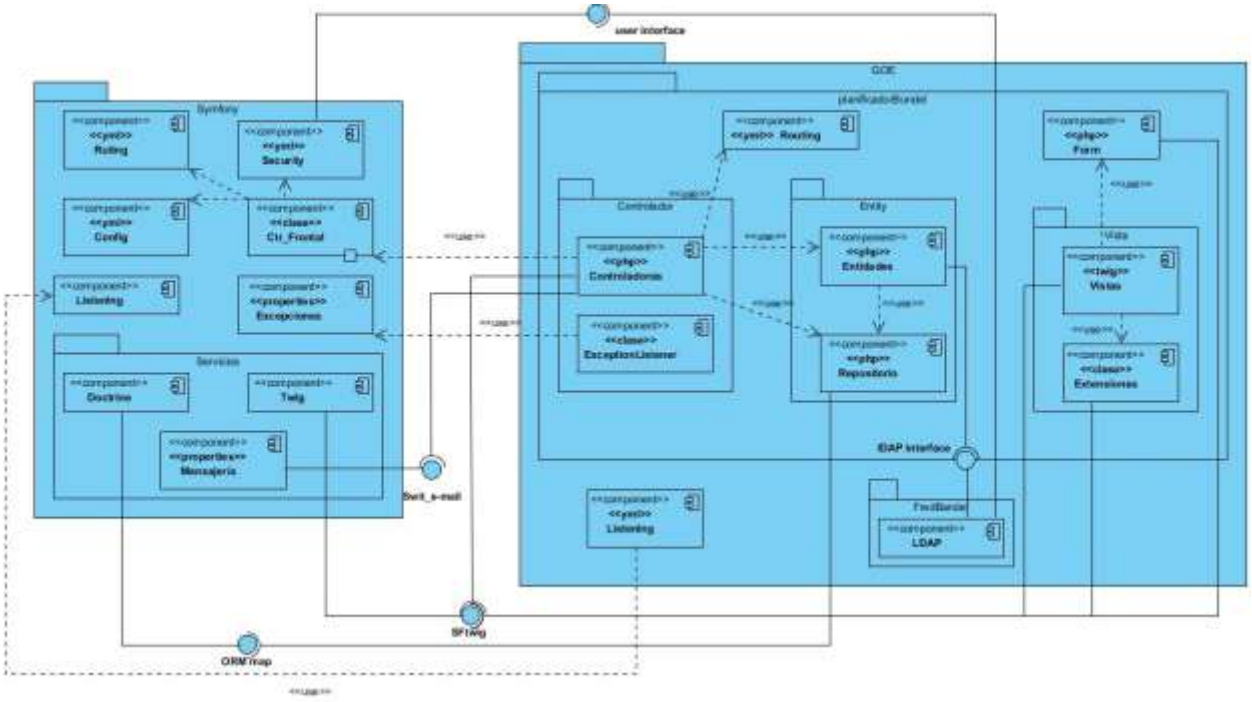


Figura 6 Diagrama de componentes

2.5.2. Diagrama de despliegue

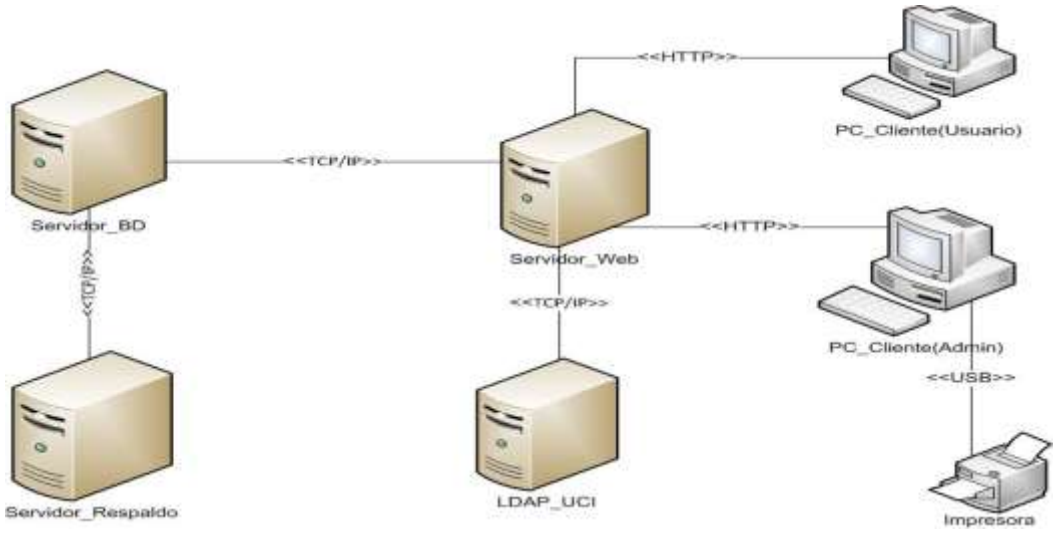


Figura 7 Diagrama de despliegue

Para el despliegue de la aplicación, se debe contar con:

Un servidor web con los módulos: php5-xsl, php5-pgsql, php5-mencache, php5-cli, php-apc, php-soap y el php5-intl.

Un servidor de Base de dato y copia de seguridad.

Pc clientes con un navegador web (Firexfox20.0 o superior, Chrome 20 o superior).

2.5.3. Estándares de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El equipo de desarrollo decidió definir varios estándares de codificación que son mostrados a continuación.

Cabecera del archivo

Siempre es importante que todos los archivos .php inicien con una cabecera específica que indique información de la versión, autor de los últimos cambios, etc. Es de cada equipo decidir si se quiere o no agregar más datos. Ejemplo:

```
/**
*@Clase persistente área. "Area.php"
*@versión:5.4.2
*@Modificado:14 de Abril del 2014
*@autor: Yasser
*/
```

Clases

\uf0d8Las clases formularios comienzan con el nombre del formulario según su función, seguido de la palabra Type (PlanificacionType.php).

\uf0d8Las clases conroladoras de negocio comienzan con el nombre seguido de la palabra Controller (PlanificacionController.php).

\uf0d8Las clases serán colocadas en un archivo .php aparte, donde sólo se colocará el código de la misma. El nombre del archivo será el mismo de la clase y siempre comenzará con mayúscula.

\uf0d8Debe colocarse un comentario antes de la declaración de la clase explicando su utilidad.

\uf0d8Los nombres de las clases se escriben con la primera letra de cada palabra que lo compone en mayúscula, haciendo uso de la notación Camello, con la variante UpperCamelCase.

Nombre de variables

\uf0d8El identificador para las variables y los parámetros serán con letras en minúsculas y en caso de ser un nombre compuesto se divide cada palabra por un signo de underscore"_".

\uf0d8 Los nombres deben ser descriptivos y concisos. No usar grandes frases, pequeñas abreviaciones para las variables.

Siempre es mejor saber qué hace una variable con sólo conocer su nombre. Esto aplica para los nombres de variables, funciones, argumentos de funciones y clases.

Para la implementación del sistema en la fase de planeación se definieron 4 iteraciones, a continuación se hace referencia a los principales aspectos de las tareas de programación realizadas en cada una de ellas.

2.5.4. Iteración 1

Al inicio de esta iteración se hace una reunión donde el equipo de desarrollo transforma el contenido de las HU en tarjetas CRC y luego en tareas de programación. La siguiente tabla muestra las tareas de programación definidas en la primera iteración donde:

Iteración: Numero de la iteración.

HU: Nombre de la Historia que va a ser implementada.

Tareas de programación: Nombre de las tareas de programación definida para una HU.

Iteración	HU	Tareas de programación
Iteración 1	Administración de áreas	Crear interfaz para la administración de áreas.
		Listar áreas
		Escribir código fuente para adicionar o editar un área
	Administración de configuraciones del área	Crear un menú para la administración de configuraciones de área.
		Crear una interfaz para cada configuración del área.
		Listar elementos de las configuraciones.
		Escribir código fuente para adicionar o modificarlas configuraciones
	Planificar guardia a estudiantes	Crear interfaz para planificar guardia
		Escoger un rango de fecha a planificar.
		Escribir código fuente para procesar los datos y planificar guardia a estudiantes

Tabla 11 Tareas de programación iteración 1

Luego de definir las tareas que serán realizadas en esta iteración, se recogen en una tabla donde se hace una descripción, para un mejor el entendimiento por parte de los

programadores. A continuación se podrán ver ejemplos de las tareas detalladas en las tablas que contienen los siguientes campos:

Numero: A cada tarea se le asigna un número para facilitar su identificación por parte del equipo de desarrollo

Número Historia: Historia a la que pertenece la tarea.

Tipo de Tareas: Se define si la tarea es de Desarrollo, Corrección o Mejora.

Programador responsable: Nombre del programador encargado de llevar a cabo la tarea.

Descripción: Especificación de lo que se debe hacer para llevar a cabo la tarea.

Tarea de programación	
Número: 1	**Número Historia**: 1
Nombre de la tarea: Crear interfaz para la administración de áreas.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntosestimados**:1
Programador responsable:	
Descripción: Crear una interfaz con los elementos que propone el administrador general. Donde se muestre una lista con las áreas existentes y se pueda adicionar o editar un área.	

Tabla 12 Descripción de la terea de programación #1

Tarea de programación	
Número: 2	**Número Historia**: 1
Nombre de la tarea: Listar áreas.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntosestimados**:1
Programador responsable:	
Descripción: Listar todas las áreas que existen en la base de datos.	

Tabla 13 Descripción de la tarea de programación #2

El resto de las tareas pueden ser consultadas en el Anexo 5.

2.5.5. Iteración **2:**

La iteración uno es terminada de forma favorable dando paso a la iteración dos donde se realiza nuevamente una reunión del equipo de trabajo y se definen las tareas correspondientes a las HU a desarrollar en esta iteración.

Iteración	Historia de usuario	Tarea de programación
Itercion2	Planificar guardia a trabajadores	Crear interfaz para planificar guardia.
		Escoger un rango de fecha a planificar.
		Escribir código fuente para procesar los datos y planificar guardia a trabajadores
	Planificar guardia a controladores	Crear interfaz para la administración de área.
		Definir código fuente para adicionar, modificar y eliminar un área.

Tabla 14 Tareas de programación iteración 2

Tarea de programación	
Número: 11	**Número Historia**: 4
Nombre de la tarea: Crear interfaz para planificar guardia.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 2
Programador responsable:	
Descripción: Crear una interfaz con los elementos que propone el administrador de área. Donde se pueda escoger un intervalo de tiempo para realizar la planificación y se pueda definir cuantas personas se planificaran por postas. También es necesario que se pueda visualizar el resultado de la planificación.	

Tabla 15 Descripción de la tarea de programación #11

Tarea de programación	
Número: 12	**NúmeroHistoria**:4
Nombre de la tarea: Escoger un rango de fecha a planificar.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 2
Programador responsable:	
Descripción:	

En esta tarea se debe implementar el código que permita al usuario escoger un rango de fecha para generar una planificación.

Tabla 16 Descripción de la tarea de programación #12

El resto de las tareas pueden ser consultadas en el Anexo 6.

2.5.6. Iteración 3

Para esta iteración se realiza el mismo procedimiento descrito en las iteraciones anteriores.

Iteración	Historia de usuario	Tarea de programación
Itercion3	Administrar planificación	Crear interfaz para la administración de planificaciones.
		Listar planificaciones
		Escribir código fuente para ver detalles o modificar un planificación
		Listar guardias asociadas a la publicación
	Evaluar usuario	Crear interfaz para la evaluar guardia.
		Listar guardias
		Escoger una fecha.
		Escribir código fuente para que se pueda emitir una evaluación.
	Dar Evaluación	Crear una interfaz para poder emitir una evaluación.
		Listar usuarios no evaluados.
		Escribir código fuente para que se pueda emitir una evaluación.
	Reportar incidencia	Crear una interfaz para reportar incidencia.
		Escribir código fuente para reportar incidencia
	Administrar incidencia	Crear una interfaz para poder administrar incidencia.
		Listar incidencias
		Escribir código fuente para adicionar, ver detalles o dar respuesta a una incidencia.
	Reportar a comisión disciplinaria o demerito	Escribir código fuente para denunciar usuarios a comisión o demerito.

		Crear una interfaz para visualizar las denuncias.

Tabla 17 Tareas de programación Iteración 3

Tarea de programación	
Número: 16	**Número Historia**: 6
Nombre de la tarea: Crear interfaz para la administración de planificaciones.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Crear una interfaz con los elementos que propone el administrador de área. Donde se muestre una lista con las planificaciones existentes en su área y se pueda ver los detalles de la planificación o modificarla.	

Tabla 18 Descripción de la tarea de programación #16

Tarea de programación	
Número: 17	**Número Historia**: 6
Nombre de la tarea: Listar planificaciones.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**:0.5
Programador responsable:	
Descripción: Listar todas las planificaciones que existen en el área.	

Tabla 19 Descripción de la tarea de programación #17

Tarea de programación	
Número: 18	**Número Historia**: 6
Nombre de la tarea: Escribir código fuente para ver detalles o modificar una planificación.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**:0.5
Programador responsable:	
Descripción: Una vez seleccionada la opción ver detalles, se debe mostrar una lista de todas las guardias asociada a la planificación, además debe permitir publicar o des publicar la planificación según sea el caso y eliminarla de desearlo.	

Tabla 20 Descripción de la tarea de programación #18

El resto de las tareas pueden ser consultadas en el Anexo 7.

2.5.7. Iteración 4

La iteración 3 fue culminada satisfactoriamente y da comienzo a la iteración4 donde se realiza el mismo procedimiento que las iteraciones anteriores.

Iteración	Historia de usuario	Tarea de programación
Iteración 4	Solicitud y cabio de guardia	Crear una interfaz para el cambio de guardia.
		Listar usuarios
		Crear código fuente para añadir un usuario
		Crear código fuente para solicitar cambio
		Crear interfaz para visualizar las solicitudes
		Crear código fuente aceptar o denegar solicitud
	Administración de cambios de guardia	Crear interfaz para la administración de cambio de guardia
		Generar código fuente para cambiar guardia
	Visualizar mis evaluaciones	Crear una interfaz para poder emitir una evaluación.
		Listar usuarios no evaluados.
		Escribir código fuente para que se pueda emitir una evaluación.
	Ajuste de planificación	Crear opciones de cambio
		Crear código fuente para ajustar la guardia
	Emitir reportes	Crear interfaz para la generación de reportes
		Crear código fuente para obtener los reportes
	Visualizar estadísticas 17	Crear interfaz para la generación de estadísticas
		Crear código fuente para obtener estadísticas

Tabla 21 Tareas de implementación Iteración 4

Tarea de programación	
Número: 34	**NúmeroHistoria**:12
Nombre de la tarea: Crear una interfaz para el cambio de guardia.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5

Programador responsable:
Descripción: Crear una interfaz que contenga, una lista de usuarios que desean cambiar guardia y otra lista con el resto de las guardias. Donde se pueda realizar una solicitud a dichos usuarios, añadirse a la lista de los que desean cambio o visualizarlas solicitudes que le han hecho.

Tabla 22 Descripción de la tarea de programación #34

Tarea de programación	
Número: 35	**NúmeroHistoria**:12
Nombre de la tarea: Listar usuarios	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Mostrar una lista de los usuarios que desean cambiar guardia y mostrar otra lista que contenga el resto de los usuarios	

Tabla 23 Descripción de la tarea de programación #35

El resto de las tareas pueden ser consultadas en el Anexo 8.

2.6. Conclusiones

En el presente capítulo se describen los artefactos, patrones y estilos de códigos a tener en cuenta en la implementación de la solución propuesta. El uso del modelo arquitectónico definido posibilitó visualizar el sistema desde diferentes puntos de vistas y mediante la correcta aplicación de los patrones de diseño, se facilitó la reutilización de código y una mejora considerable del nivel de complejidad de las clases. La correcta aplicación de la metodología seleccionada acorto el tiempo de desarrollo estimado.

CAPÍTULO 3: VALIDACIÓN DE LOS RESULTADOS

3.1 Introducción

Durante el desarrollo de este capítulo se presentarán los resultados obtenidos durante la fase de prueba, se estarán viendo los resultados de las pruebas de aceptación que fueron hechas con el objetivo de aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También se realizan las pruebas unitarias encargadas de verificar el código

3.2 Pruebas funcionales o de aceptación

A continuación se muestran los casos de pruebas diseñados a las historias de usuario Planificar guardia a estudiante y Emitir evaluación debido a que las mismas incluyen el conjunto de funcionalidades críticas para el desarrollo exitoso del sistema informático propuesto en la investigación.

Dichos casos de pruebas se describirán en tablas que contendrán los siguientes campos:

Código: Identificador de la prueba realizada, a su vez será sugerente al nombre de la prueba a la que hace referencia.

Historia de usuario: Nombre de la HU a la que hace referencia la prueba a realizar.

Nombre: Nombre de la prueba

Descripción de la prueba: Breve descripción de la prueba.

Condiciones de Ejecución: Muestra las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.

Entradas / Pasos de Ejecución: Descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.

Resultados de la prueba: Descripción breve del resultado que se espera obtener con la prueba realizada.

La evaluación de la prueba realizada se hará según el resultado de la misma, la tendrá uno de los tres resultados que a continuación se describen:

Satisfactoria: Cuando el resultado de la prueba es exactamente el esperado por el usuario.

Parcialmente bien: Cuando el resultado no es completamente el esperado por el cliente o usuario de la aplicación y muestra resultados erróneos o fuera de contexto.

Mal: Cuando el resultado de la prueba realizada genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contexto, trayendo como consecuencia que la funcionalidad requerida por el cliente no tenga resultado, lo que invalida también la HU.

El siguiente caso de prueba corresponde a la funcionalidad "Planificar guardia a estudiante" encargada de realizar una distribución de recursos para obtener así una planificación de la guardia:

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario**: Planificar guardia a estudiantes
Nombre: Planificar Estudiantes.	
Descripción: Probar que se realice la planificación de los usuarios de tipo estudiantes de forma correcta.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración. Se debe permitir la selección de un rango de tiempo a planificar.	
Entrada / Pasos de Ejecución: Se selecciona la fecha de inicio y fin. Se genera una distribución, la cual puede ser ajustada. Se asignan los controladores y se publica la planificación.	
Resultado Esperado: Los usuarios de tipo estudiante son distribuidos en postas y turnos de guardia dentro del rango de fechas seleccionado.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 24 Caso de prueba Planificar guardia a estudiantes

Caso de prueba correspondiente a la funcionalidad "Emitir evaluación" encargada de evaluar a los implicados en el proceso de guardia:

Caso de Prueba de Aceptación	
Código: HU7_P1	Historia de Usuario: Emitir evaluación
Nombre: Emitir evaluación.	
Descripción: Probar que se emita la evaluación de la guardia de un usuario.	

Condiciones de Ejecución: La aplicación debe ejecutarse con privilegios de controlador. Se debe de tener guardias planificadas.
Entrada / Pasos de Ejecución: Se selecciona una guardia, y se le asigna una evaluación.
Resultado Esperado: Se emite una evaluación de una guardia determinada.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 25 Caso de prueba Emitir evaluación

El resto de los casos de pruebas asociados a las restantes historias de usuarios pueden consultarse en el anexo 4.

Análisis de los resultados:

Para validar que la salida emitida por el sistema informático concordara con el resultado esperado por el cliente se diseñaron en la primera iteración6 pruebas funcionales en conjunto cliente-desarrolladores para las 3 Historias de usuario que se implementaron en esa iteración de las cuales 5 salidas coincidieron con los resultados esperados representando un 83%. Fue notable el poco tiempo empleado por el sistema en la devolución del resultado, por otra parte 1 prueba resultó fallida representando un 17% pero rápidamente se descubrió la causa y en poco tiempo fue corregida, en una segunda iteración. De 10 pruebas funcionales realizadas 10 resultaron satisfactorias constituyendo un 100% lo que significó un logro para el equipo, mientras que en una tercera iteración se realizaron 11 pruebas que representan el 72% quedando dos pruebas insatisfactorias lo que representa el 28%. En la última iteración se arrojó resultados satisfactorios pues para el desarrollo de 17 pruebas funcionales se obtuvieron un total de 17 pruebas exitosas para un 100% de pruebas satisfactorias.

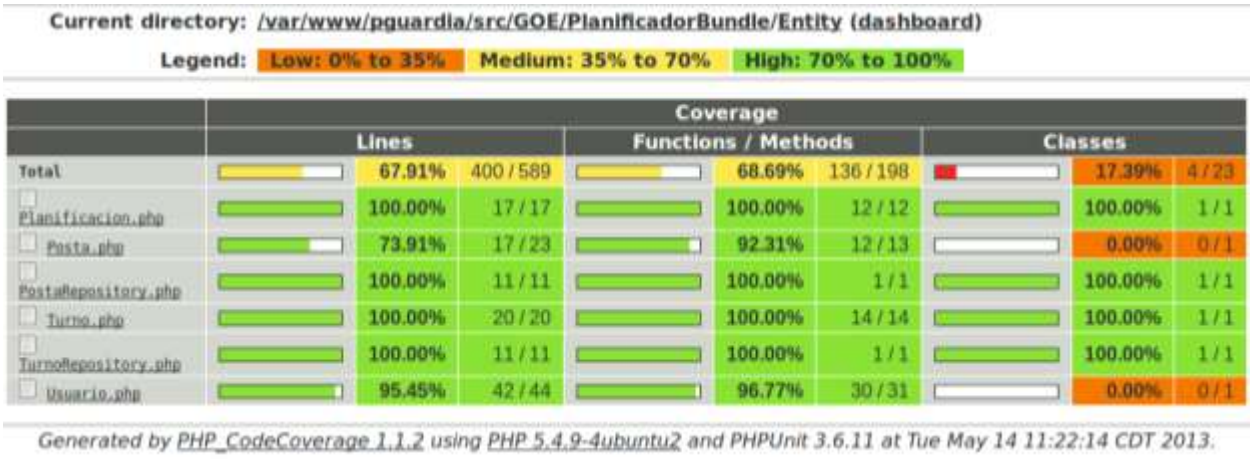
A continuación se muestra como quedan reflejados los resultados por cada una de las iteraciones de pruebas funcionales realizadas el sistema:

Iteración	Satisfactoria	Insatisfactoria	Total
1	5	1	6
2	7	0	7
3	8	2	11
4	17	0	17

Tabla 26 Iteraciones de pruebas funcionales

3.3 Pruebas unitarias

A continuación se muestran un resumen generado por el plugin Selenium para NetBeans acerca de las pruebas unitarias realizadas al código. En el mismo se evidencia un 70% de completado lo que representa un 100% de las clases implementadas. No se tienen en cuenta el código propio del FrameWork ni los nomencladores[11] (no se tiene en cuenta este sub conjunto del domino de las clases por no ser persistido por el usuario, debido a que el mismo está comprendido dentro de la carga inicial del sistema).



Current directory: /var/www/pguardia/src/GOE/PlanificadorBundle/Entity (dashboard)

Legend: Low: 0% to 35% Medium: 35% to 70% High: 70% to 100%

	Coverage							
	Lines			Functions / Methods			Classes	
Total	67.91%	400 / 589		68.69%	136 / 198		17.39%	4 / 23
Planificacion.php	100.00%	17 / 17		100.00%	12 / 12		100.00%	1 / 1
Posta.php	73.91%	17 / 23		92.31%	12 / 13		0.00%	0 / 1
PostaRepository.php	100.00%	11 / 11		100.00%	1 / 1		100.00%	1 / 1
Turno.php	100.00%	20 / 20		100.00%	14 / 14		100.00%	1 / 1
TurnoRepository.php	100.00%	11 / 11		100.00%	1 / 1		100.00%	1 / 1
Usuario.php	95.45%	42 / 44		96.77%	30 / 31		0.00%	0 / 1

Generated by PHP_CodeCoverage 1.1.2 using PHP 5.4.9-4ubuntu2 and PHPUnit 3.6.11 at Tue May 14 11:22:14 CDT 2013.

3.3.1 Tamaño operacional de la clase (TOC)

El grado de calidad y fiabilidad del diseño alcanzado por el sistema, se midió a través de métricas basadas en las clases; las cuales miden categorías como tamaño operacional de clase y relaciones entre clases, permitiendo el resultado de ambas métricas validar el diseño propuesto en la investigación.

A continuación se muestra como se aplicó la métrica Tamaño Operacional de Clase (TOC) al sistema informático. El tamaño operacional de una clase se puede determinar empleando medidas para saber el número total de operaciones.

Las clases que juegan un papel importante en los procesos del sistema informático propuesto en la investigación se encuentran relacionadas en el controlador y en el modelo, siendo aplicada a estas clases la métrica TOC.

[11]Representación abstracta de una clase la cual no es persistida por el usuario.

A continuación se muestran las clases que juegan un papel fundamental en los procesos principales del sistema informático:

Clase	Cantidad de Procedimientos
Área	8
Posta	7
Guardia	7
Usuario	11
Turno	7
Guevaluacion	8
Incidencia	8
Planificación	15
SolicitudDeCambio	13

Tabla 27 Clases fundamentales

Resultados obtenidos

Los umbrales que referencian a los atributos de calidad: responsabilidad de las clases, complejidad al implementar las mismas, así como sus niveles de reutilización, los cuales se le aplican a las 9 clases del sistema informático con un promedio de 8.3 operaciones, quedan especificados en la siguiente tabla:

	Categoría	Criterio
Responsabilidad	Baja	< =9.3
	Media	Entre 9.3 y 18.6
	Alta	> 18.6
Complejidad de implementación	Baja	< =9.3
	Media	Entre 9.3 y 18.6
	Alta	> 18.6
Reutilización	Baja	>18.6

	Media	Entre 9.3 y 18.6
	Alta	<= 9.3

Tabla 28 Umbrales para la responsabilidad, complejidad y reutilización.

En la Tabla y las figuras siguientes se muestran cómo quedan reflejados los atributos de calidad antes mencionados en las clases sometidas a la métrica de diseño propuesta:

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
Área	8	Baja	Baja	Alta
Posta	7	Baja	Baja	Alta
Guardia	7	Baja	Baja	Alta
Usuario	11	Media	Media	Media
Turno	7	Baja	Baja	Alta
Guevaluacion	8	Baja	Baja	Alta
Incidencia	8	Baja	Baja	Alta
Planificación	15	Media	Media	Media
SolicitudDeCambio	13	Media	Media	Media

Tabla 29 Clases analizadas con resultados obtenidos.

Figura 8 Atributo de calidad que refleja la responsabilidad.

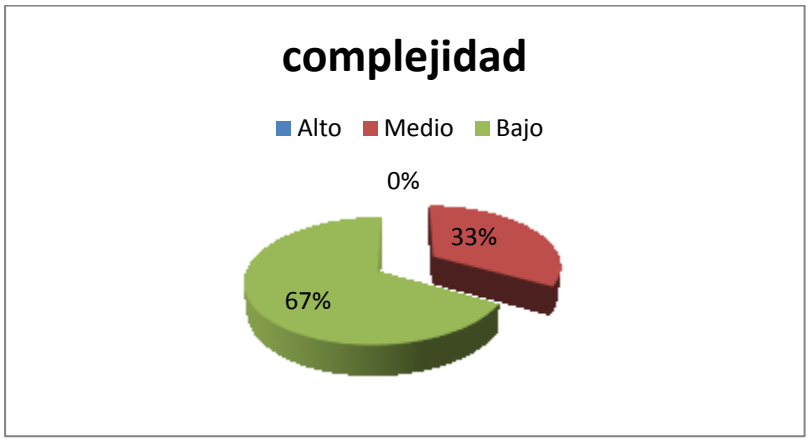


Figura 9 Atributo de calidad que refleja la complejidad.

Figura 10 Atributo de calidad que refleja la reutilización.

Tomando como referencia la información referida por las Figuras 10, 11 y 12 se muestra con un 67% al umbral bajo como el porciento sobresaliente a los atributos de responsabilidad y complejidad y con un 67% al umbral alto al atributo reutilización.

3.4 Conclusiones.

En este capítulo quedan descritos los artefactos necesarios para llevar acabo las pruebas del sistema. La aplicación de las pruebas de aceptación al final de cada iteración y las revisiones durante todo el desarrollo, posibilitaron la corrección de posibles errores y la obtención de un prototipo funcional acorde a los requisitos definidos. Los resultados de cada prueba, incluyendo las unitarias, generaron una retroalimentación permanente, mostrándoles a los desarrolladores la calidad de su trabajo.

CONCLUSIONES GENERALES

Sobre la base del análisis, interpretación y sistematización de las indagaciones empíricas y teóricas, a continuación se presentan las siguientes conclusiones de la investigación:

- El estudio teórico de los presupuestos actuales en los que se sustenta la gestión de planificación, procesos de planificación de la GOE en Cuba y soluciones informáticas que se han desarrollado en otros países, permitió a los autores de esta investigación fundamentar la necesidad de mejorar la gestión de la planificación del proceso de la GOE de la UCI y adquirir conocimientos que posteriormente se aplicaron en el desarrollo de la solución.

- La documentación resultante de la presente investigación, cuenta con todos los elementos necesarios para ser consultada en caso de que se desee agregar nuevas funcionalidades.

- Se demuestra, con la obtención del prototipo funcional, que el sistema reduce al mínimo los errores en la planificación, lo cual disminuye las inconformidades y aumenta la confiabilidad en los informes estadísticos de esta área. Así como se disminuyó el tiempo para llevar a cabo este proceso.

- La validación de la propuesta, demuestra la viabilidad, pertinencia y factibilidad en la práctica, posibilitó el cumplimiento de los objetivos propuestos en el presente trabajo de diploma, lo cual conlleva a un cumplimiento del objetivo general.

RECOMENDACIONES

BIBLIOGRAFÍA

1. **CONSEJO DE MINISTROS.** *Gaceta Oficial.* LA HABANA : s.n., 2007. 021.

2. **Zayas, Carlos Alvarez de.** *METODOLOGIA DE LA INVESTIGACION CIENTIFICA.* 1995.

3. **Pall, Gabriel A.** *Quality Process Management.* Mexico : Prentice Hall, 1987.

4. **Heras, Miguel A.** *Gestión de la producción.* Barcelona : ESADE, 1996.

5. **Torres, Luis Cabrera.** *Procedimiento de análisis y mejoramiento de procesos. Aplicación a una corporación comercial.* 2003.

6. **Cooper, Robin.** *When Lean Enterprises Collide: Competing through Confrontation.* 1995.

7. **JURAN, J. M.** *Jurán y la planificación para la calidad.* s.l. : Díaz de Santos, 1990.

8. **Facultad de Ciencias Exactas (UNICEN).** Facultad de Ciencias Exactas (UNICEN). *sitio web de la Facultad de Ciencias Exactas (UNICEN).* [En línea] 2003. [Citado el: 24 de noviembre de 2012.] http://www.exa.unicen.edu.ar/catedras/modemp/03_Introduccion_Integracion_Modelado.pdf.

9. **Fong, Miguel Chin.** Galeon.com Hispavista. *sitio web Galeon.com.* [En línea] [Citado el: 25 de Noviembre de 2012.] www.galeon.com/mcf1/plafinal.htm.

10. **GITMAN, LAWRENCE J.** *Fundamentos de administración financiera.* Mexico : Oxford University Press, 1997.

11. **SAINZ DE VICUÑ ANCÍN, J. M.** *El plan de marketing en la práctica.* Madrid : ESIC Editorial, 1999.

12. **Roncajolo, Fernando A. Gabaldon.** *Gerencias de organizaciones de servicio.* Venezuela : Consejo de publicaciones de la Universidad de los Andes, 2001.

13. **Chaín Navarro, Celia.** Introducción a la gestión y análisis de recursos de información en ciencia y tecnología. 1995.

14. **Cea, Adrián Mendoza.** Planifica. *sites.google.com.* [En línea] [Citado el: 01 de 01 de 2013.] https://sites.google.com/site/profesorangol/home.

15. **intelektron.** Kit Control de Rondas. *intelektron.* [En línea] intelektron. [Citado el: 1 de Enero de 2013.] http://www.intelektron.com.ar.

16. **Almeida, Humberto y Torrez Aguilera, Grettell.** *Gestor Web para el control de la Guardia Obrera de la Universidad de las Ciencias Informaticas (UCI).* 2009.

17. **Davila Perez, Yuniesky y Martinez Padron, Yadrian.** *Sistema de Gestion de Informacion de la Facultad 8. Modulo para la Gestion de la Residencia Estudiantil Version 2.* 2008.

18. **Rosabal Sanchez, Yisel.** *Sistema de gestion de informacion del Area de Extension Universitaria.*

19. **Gonzalez, Benjamin.** http://codigolinea.com/. [En línea] 2008. [Citado el: 03 de abril de 2013.] http://codigolinea.com/2008/06/04/compararcion-y-rendimiento-de-frameworks-php/.

20. **Rius Diaz, Francisca.** *Bioestadistica: Metodos y Aplicaciones.* Malaga : Universidad de Malaga.

21. **Boyarynov, Eugene.** *Symfony2 The Book.* 2011.

22. **Symfony. Open-Source PHP Web marco de trabajo . [En línea] http://www.symfony-project.org/.**

23. Holscher, Eric, Leifer, Charles y Grace, Bobby. Doctrine Documentation. [En línea] 2010. [Citado el: 3 de Enero de 2013.] http://www.doctrine-project.org/documentation/manual/1_0/en/introduction-to-models.

24. Introducción a JavaScript. *Introducción a JavaScript.* 2009.

25. Bowen, Rich. *What's New in Apache Web Server 2.2?* s.l. : O'Reilly Media, Inc., 2007.

26. Bogus, Andre. *Lighttpd.* s.l. : Packt Publishing Ltd, 2008.

27. Adam, Kelli y Stanley, Lisa. *Internet Information Services Administration.* s.l. : Sams Publishing, 2000.

28. Russell, Jesse y Cohn, Ronald. *Thttpd.* s.l. : Book on Demand, 2012.

29. Gutiérrez Gallardo, Juan Diego. *MySQL.* s.l. : Grupo Anaya Comercial, 2004.

30. Stinson, Barry. *PostgreSQL: Essential Reference.* s.l. : Sams Publishing, 2001.

31. Firebird. Firebird. [En línea] 2012. [Citado el: 3 de Enero de 2013.] http://www.firebirdsql.org.

32. Zaragoza, Mtra. María de Lourdes Santiago. Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado (RUP). [En línea] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

33. Beck, Kent. Planning Extreme Programming. s.l. : Addison Wesley, 2000.

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

35. Wells, Don. The Rules and Practices of Extreme Programming. [En línea] [Citado el: 2 de Febrero de 2013.] http://www.extremeprogramming.org/rules.html.

36. Noble Martin, Angela y Biddle James, Robert. The XP Customer Role in Practice: Three Studies Agile. 2004.

37. Kniberg, H. Scrum and XP from the Trenches. 2008.

38. Definición ABC. [En línea] [Citado el: 18 de enero de 2012.] http://www.definicionabc.com/general/herramienta.php.

39. netbeans sitio web. Netbeans. *netbeans sitio web.* [En línea] [Citado el: 4 de Enero de 2013.] http://netbeans.org.

40. Kruchtens, Philippe. Architectual Blueprints – the "4+1" View Model of Software Architecture. *Computer Science at UBC.* [En línea] http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf.

41. 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. *IEEE Standards Association.* [En línea] http://standards.ieee.org/findstds/standard/1471-2000.html.

42. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. ISBN 970-17-0261-1.

43. Symfony. [En línea] [Citado el: 12 de noviembre de 2012.] http://www.symfony.com/.

44. Symfony en español. [En línea] 12 de marzo de 2013. http://gitnacho.github.com/symfony-docs-es/book/doctrine.html.

45. Untis Teem. Untis. [En línea] Untis Express. [Citado el: 1 de Enero de 2013.] http://www.grupet.at/.

46. AntamediaMedical. Corporación AntamediaMedical. *Sitio web AntamediaMedical.* [En línea] AntamediaMedical. [Citado el: 2 de Enero de 2013.] http://www.antamediamedical.com.

47. Aragonés López, Ing. Carlos, y otros. informatica2007. *sitio web de informatica2007.sld.cu.* [En línea] http://www.informatica2007.sld.cu/Members/aragones/sistema-automatizado-para-la-atencion-medica-integral-a-pacientes-vih-sida-201csidatrat201d/.

48. CEDISAP. SLD.CU. *Sitio web de Salud.* [En línea] CEDISAP. INFORMATICA MEDICA . [Citado el: 2 de Enero de 2013.] http://www.sld.cu/instituciones/cedisap/Medclien.htm.

49. Guilarte, Darien Turcaz. *Sistema automatizado para la planificación de Turnos y gestión de reportes estadísticos en el Cited.* Facultad de Ingeniería Informática, Instituto Superior Politécnico "José Antonio Echeverría" (CUJAE). La Habana : s.n., 2003. Trabajo de diploma para optar por el título de Ingeniería en Informática.

50. Holzschlag, Molly E. y Pellerin, Jason. *Perl: Web Site Workshop.* s.l. : Sams Publishing, 2002.

51. Holden, Steve. *Python Web Programming.* Washinton : Sams Publishing, 2002.

52. PHP Documentation Group. *Manual de PHP.* s.l. : PHP Documentation Group, 2012.

53. Wall, Larry, Christiansen, Tom y Orwant, Jon. Programming Perl. Tercera. 2000, 18.

54. Instituto Andaluz de Tecnología. *Guía para una gestión basada en procesos.* s.l. : Instituto Andaluz de Tecnología (IAT), 2005.

55. Symfony users. [En línea] http://groups.google.com/group/symfony-users/msg/cd94d2ddb2057355.

56. Sitio oficial de PHP. [En línea] http://www.php.net/.

57. Prieto, Félix. Patrones de diseño. [En línea] http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf..

58. Patrones del "Gand of Four". Madrid, España : s.n.

59. Doctrine Documentation. [En línea] http://www.doctrine-project.org/documentation/manual/1_0/en/introduction-to-models.

60. Roger S. Pressman. *Ingeniería de Software. Un enfoque práctico.*

61. Pressman. *Ingeniería del Software: Un Enfoque Práctico.* Sexta Edición. 2005. pág. 900 .

62. Introducción a Redes. Arquitectura Cliente/Servidor. *Microsoft Windows Server.* [En línea] http://www.juansa.net/Admin2003/cliser.htm.

63. *Metodologías Ágiles o formales y robustas.* Pérez, Dr.C Profesor Auxiliar Pedro Y. Piñero Pérez y Dr.C Profesor Auxiliar Pedro Y. Piñero.

64. Craig, LARMAN. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México: PRENTICE HALL : s.n., 1999.

65. Larman, G. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Prentice Hal : s.n., 2004.

66. (traducción), Nacho Pacheco. *Symfony2-es.* 2011.

ANEXO

Anexo 1 Fichas de historias de usuarios.

Tabla 30Historia de usuario Administración de área.

Historia de Usuario	
Número: 1	Usuario: Administrador general.
Nombre de Historia de Usuario: Administración de área	
Programador:	
Prioridad en Negocio: Alta (Alta/Media/Baja)	Puntos Estimados: 3
Riesgo en Desarrollo: Bajo (Alto/Medio/Bajo)	Iteración Asignada:1
Descripción: El administrador general de la aplicación adiciona o elimina un área. Para esto es necesario definir el nombre del área.	
Observaciones:	

Tabla 31Historia de usuario Administración de configuraciones del área

Historia de Usuario	
Número: 2	Usuario: Administrador de área.
Nombre de Historia de Usuario: Administración de configuraciones del área	
Programador:	
Prioridad en Negocio: Alta	Puntos Estimados: 3

(Alta/Media/Baja)	
Riesgo en Desarrollo: Medio (Alto/Medio/Bajo)	Iteración Asignada:1
Descripción: Inicia cuando el Administrador de área una desea adicionar, modificar o eliminar usuarios, postas, turnos, y roles que existen en el área.	
Observaciones:	

Tabla 32Historia de usuario Planificar guardia a estudiantes

Historia de Usuario	
Número: 3	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Planificar guardia a estudiantes	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 8
Riesgo en Desarrollo: Alto **(Alto/Medio/Bajo)**	**Iteración Asignada:**1
Descripción: Inicia cuando el Administrador de área decide planificar la guardia a los estudiantes en su área. Para esto necesita el listado de estudiantes, de postas y turnos con los que dispone y definir un rango de tiempo a planificar.	
Observaciones:	

Tabla 33Historia de usuario Planificar guardia a trabajadores

Historia de Usuario	
Número: 4	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Planificar guardia a trabajadores	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 8

Riesgo en Desarrollo: Alto **(Alto/Medio/Bajo)**	**Iteración Asignada:**2
Descripción: Inicia cuando el Administrador de área decide planificar la guardia a los trabajadores en su área. Para esto necesita el listado de trabajadores, de postas y turnos con los que dispone y definir un rango de tiempo a planificar.	
Observaciones:	

Tabla 34Historia de usuario Planificar guardia a controladores

Historia de Usuario	
Número: 5	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Planificar guardia a controladores	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 6
Riesgo en Desarrollo: Alto **(Alto/Medio/Bajo)**	**Iteración Asignada:**2
Descripción: Inicia cuando el Administrador de área una vez planificada la guardia a estudiantes o trabajadores desea continuar y asigna a los controladores a las guardias existentes en una planificación.	
Observaciones:	

Tabla 35Historia de usuario Administrar planificación.

Historia de Usuario	
Número:6	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Administrar planificación.	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio **(Alto/Medio/Bajo)**	**Iteración Asignada:**3
Descripción: Inicia cuando el administrador de área desea publicar o eliminar las planificaciones realizadas. Para esto es necesario que pueda visualizar las planificaciones existentes en su área.	

Observaciones:

Tabla 36Historia de usuario Evaluar guardias

Historia de Usuario	
Número: 7	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Evaluar guardias	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada:**3
Descripción: Inicia cuando el Administrador de área decide evaluar una guardia. Para esto necesita definir una fecha listando así las guardias correspondientes donde puede modificar la evaluación a un usuario o evaluar al controlador.	
Observaciones:	

Tabla 37Historia de usuario Dar evaluación

Historia de Usuario	
Número: 8	**Usuario:** Controlador.
Nombre de Historia de Usuario: Dar evaluación	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada:**3
Descripción: Inicia cuando el controlador decide evaluar una guardia. Para esto necesita tener la lista de las guardias que controla y no ha evaluado aún.	
Observaciones:	

Tabla 38: **Historia de usuario Reportar incidencias.**

Historia de Usuario

Número: 9	**Usuario:** Controlador.
Nombre de Historia de Usuario: Reportar incidencias.	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada: 3**
Descripción: Inicia cuando un controlador necesita reportar una incidencia. Para esto es necesario poder realizar una descripción de la misma.	
Observaciones:	

Tabla 39Historia de usuario Administrar incidencias.

Historia de Usuario	
Número: 10	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Administrar incidencias.	
Programador:	
Prioridad en Negocio: Alta **(Alta/Media/Baja)**	**Puntos Estimados:** 3
Riesgo en Desarrollo: Medio **(Alto/Medio/Bajo)**	**Iteración Asignada:**3
Descripción: Inicia cuando un administrador de área desea ver las incidencias en su área donde puede darle respuesta o ver sus detalles así como eliminarla. Para esto es necesario visualizar la descripción de la misma y poder escribir una respuesta.	
Observaciones:	

Tabla 40Historia de usuario Reportar a comisión disciplinaria y demerito.

Historia de Usuario	
Número: 11	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Reportar a comisión disciplinaria y demerito.	
Programador:	
Prioridad en Negocio: Media **(Alta/Media/Baja)**	**Puntos Estimados:** 3

Riesgo en Desarrollo: Alto **(Alto/Medio/Bajo)**	**Iteración Asignada:**3
Descripción: Inicia cuando un administrador de área desea ver los reportados en su área a comisión disciplinaria o demerito. Para esto es necesario mostrar una lista con las personas que sean evaluados de mal dos veces seguidas.	
Observaciones:	

Tabla 41Historia de usuario Solicitud y cabio de guardia.

Historia de Usuario	
Número: 12	**Usuario:** Usuario.
Nombre de Historia de Usuario: Solicitud y cabio de guardia.	
Programador:	
Prioridad en Negocio: Media **(Alta/Media/Baja)**	**Puntos Estimados:** 3
Riesgo en Desarrollo: Medio **(Alto/Medio/Bajo)**	**Iteración Asignada:**4
Descripción: Inicia cuando un usuario desea cambiar su guardia. Para esto es necesario que pueda publicar su deseo de cambiar, solicitar el cambio a otro usuario o responder a solicitudes de cambio que le sean hachas.	
Observaciones:	

Tabla 42Historia de usuario Administración de cambios de guardia.

Historia de Usuario	
Número: 13	**Usuario:** Usuario.
Nombre de Historia de Usuario: Administración de cambios de guardia.	
Programador:	
Prioridad en Negocio: Media **(Alta/Media/Baja)**	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada:**4
Descripción: Inicia cuando el administrador de área desea cambiar la guardia a dos usuarios determinados. Para esto es necesario que pueda visualizar las guardias de su área.	

Observaciones:

Tabla 43 Historia de usuario Visualizar mis evaluaciones

Historia de Usuario	
Número: 14	**Usuario:** Usuario.
Nombre de Historia de Usuario: Visualizar mis evaluaciones	
Programador:	
Prioridad en Negocio: Baja **(Alta/Media/Baja)**	**Puntos Estimados: 2**
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada:**4
Descripción: Inicia cuando un usuario decide revisar sus evaluaciones. Para esto necesita tener la lista de las guardias que ha realizado y la evaluación asignada.	
Observaciones:	

Tabla 44Historia de usuario Ajuste de planificación.

Historia de Usuario	
Número: 15	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Ajuste de planificación.	
Programador:	
Prioridad en Negocio: Baja **(Alta/Media/Baja)**	**Puntos Estimados:** 3
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada:**4
Descripción: Inicia cuando una vez realizada una planificación, el administrador de área realiza modificaciones a esta bajo su responsabilidad. Para esto es necesario que pueda visualizar las guardias de la planificación.	
Observaciones:	

Tabla 45 Historia de usuario Emitir reportes.

Historia de Usuario	
Número: 16	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Emitir reportes.	
Programador:	
Prioridad en Negocio: Baja **(Alta/Media/Baja)**	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada:**4
Descripción: Inicia cuando el administrador de área desea emitir reportes.	
Observaciones:	

Tabla 46Historia de usuario Visualizar estadísticas.

Historia de Usuario	
Número: 17	**Usuario:** Administrador de área.
Nombre de Historia de Usuario: Visualizar estadísticas.	
Programador:	
Prioridad en Negocio: Baja **(Alta/Media/Baja)**	**Puntos Estimados:** 2
Riesgo en Desarrollo: Bajo **(Alto/Medio/Bajo)**	**Iteración Asignada:**4
Descripción: Inicia cuando el administrador de área desea ver el comportamiento y funcionamiento de la guardia en su área.	
Observaciones:	

Anexo 2 Tarjetas CRC

Tabla 47Tarjeta CRC: Área

Clase Área	
Responsabilidad	**Colaboración**

Crear Eliminar Editar Listar Detallar Actualizar	

Tabla 48Tarjeta CRC: Guardia

Clase Guardia	
Responsabilidad	**Colaboración**
Crear Eliminar Editar Listar Detallar Actualizar	Posta Turno

Tabla 49Tarjeta CRC: Guevaluacion

Clase Guevaluacion	
Responsabilidad	**Colaboración**
Crear Eliminar Editar Listar Detallar Actualizar Evaluar	Guardia Usuario Evaluacion

Tabla 50Tarjeta CRC: Incidencia

Clase Incidencia	
Responsabilidad	**Colaboración**
Crear Eliminar Editar Listar Detallar Actualizar Responder	Área Usuario

Tabla 51Tarjeta CRC: Posta

Clase Posta	
Responsabilidad	**Colaboración**
Crear Eliminar Editar Listar Detallar Actualizar	Área Tipo

Tabla 52Tarjeta CRC: Turno

Clase Turno	
Responsabilidad	**Colaboración**

Crear Eliminar Editar Listar Detallar Actualizar	Área Tipo Día

Tabla 53Tarjeta CRC: SolicitudDeCambio

Clase SolicitudDeCambio	
Responsabilidad	**Colaboración**
Crear Eliminar Editar Listar Detallar Actualizar Iniciar Solicitud Cancelar Solicitud	Área Guardia Usuario

Tabla 54Tarjeta CRC: Usuario

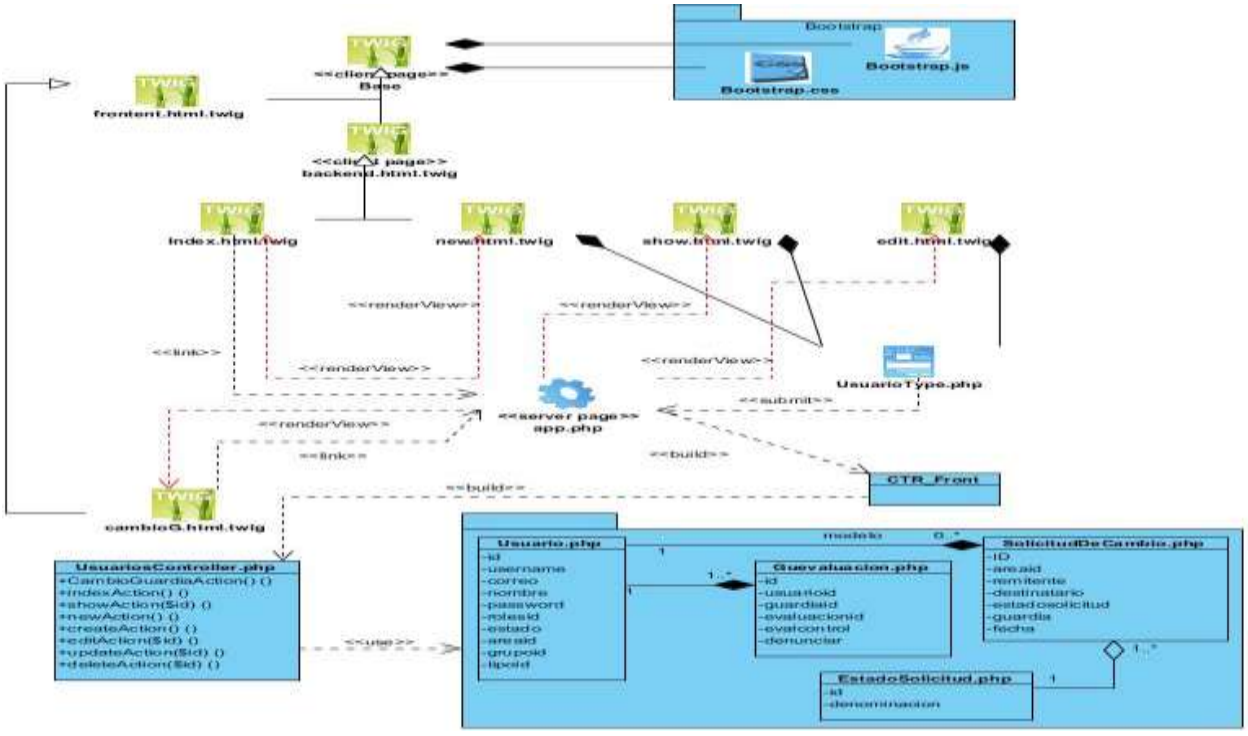
Clase Usuario	
Responsabilidad	**Colaboración**
Crear Eliminar Editar Listar Detallar Actualizar	Área Roles Tipo Estado Grupo

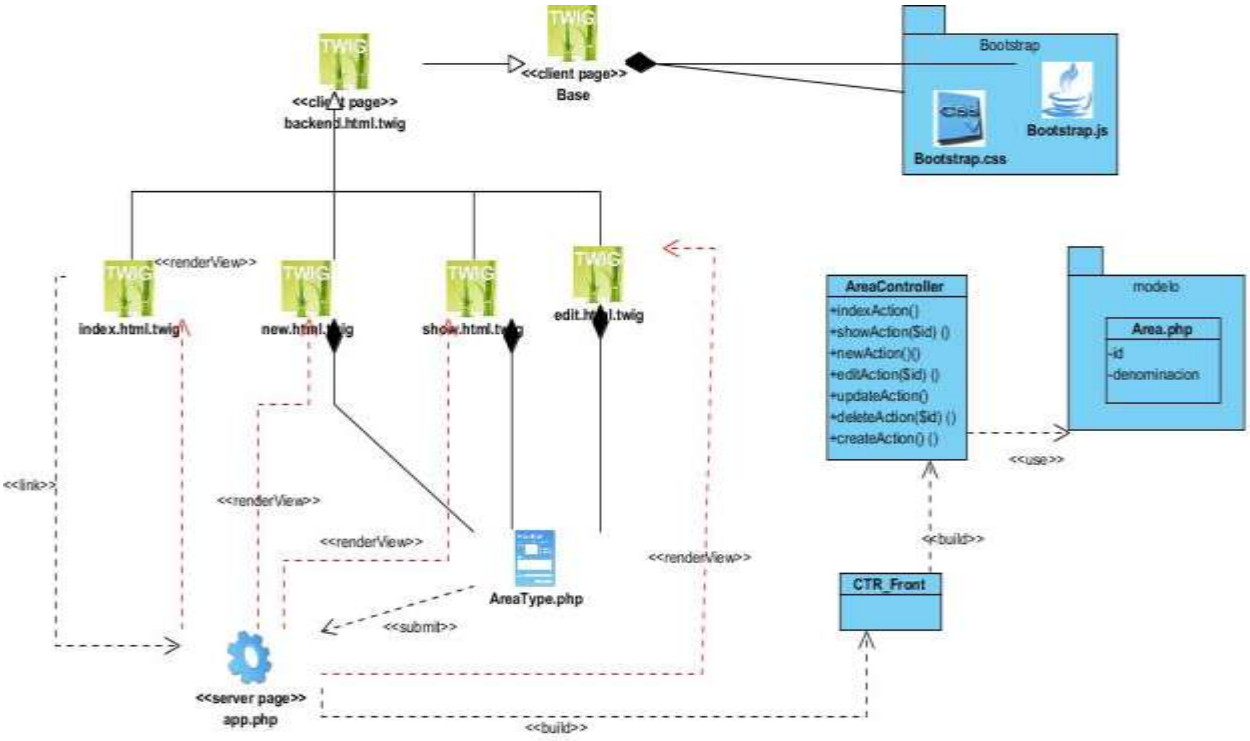
Ajustar Cambio de Guardia	

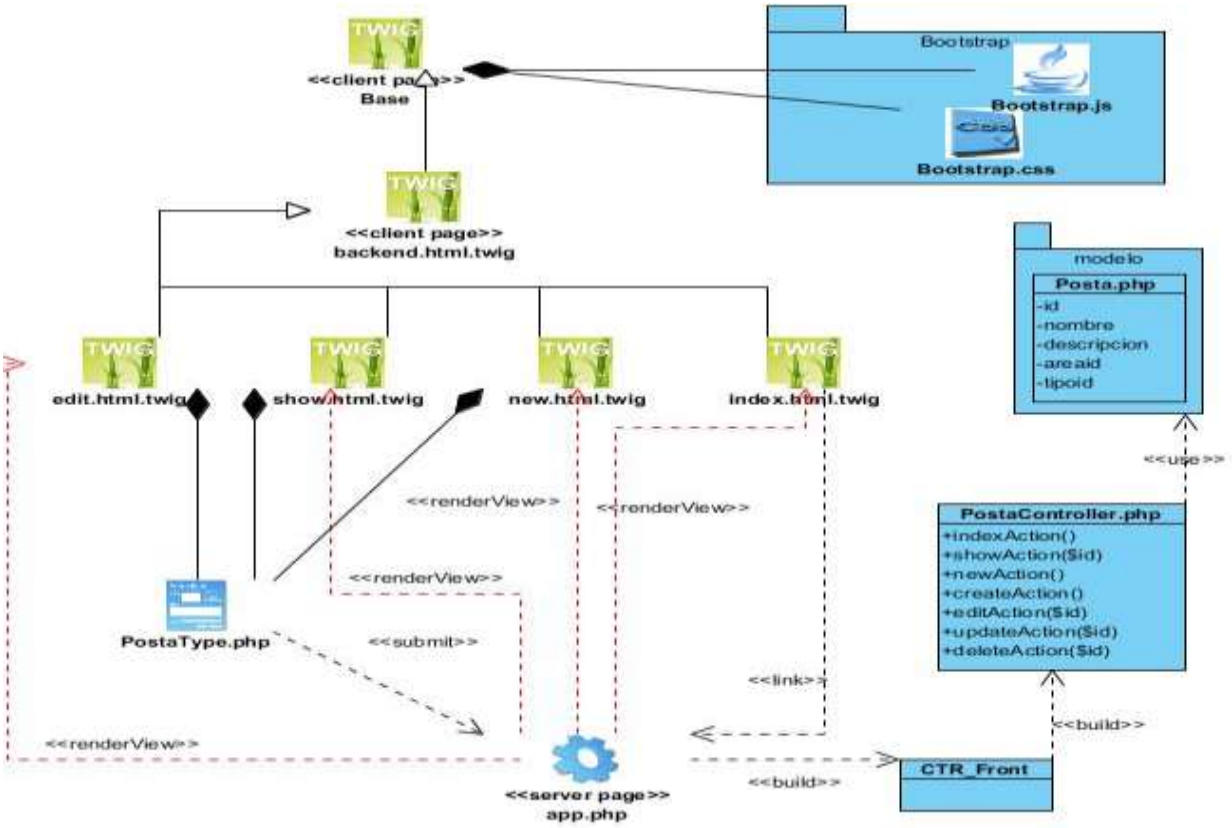
Tabla 55Tarjeta CRC: Traza

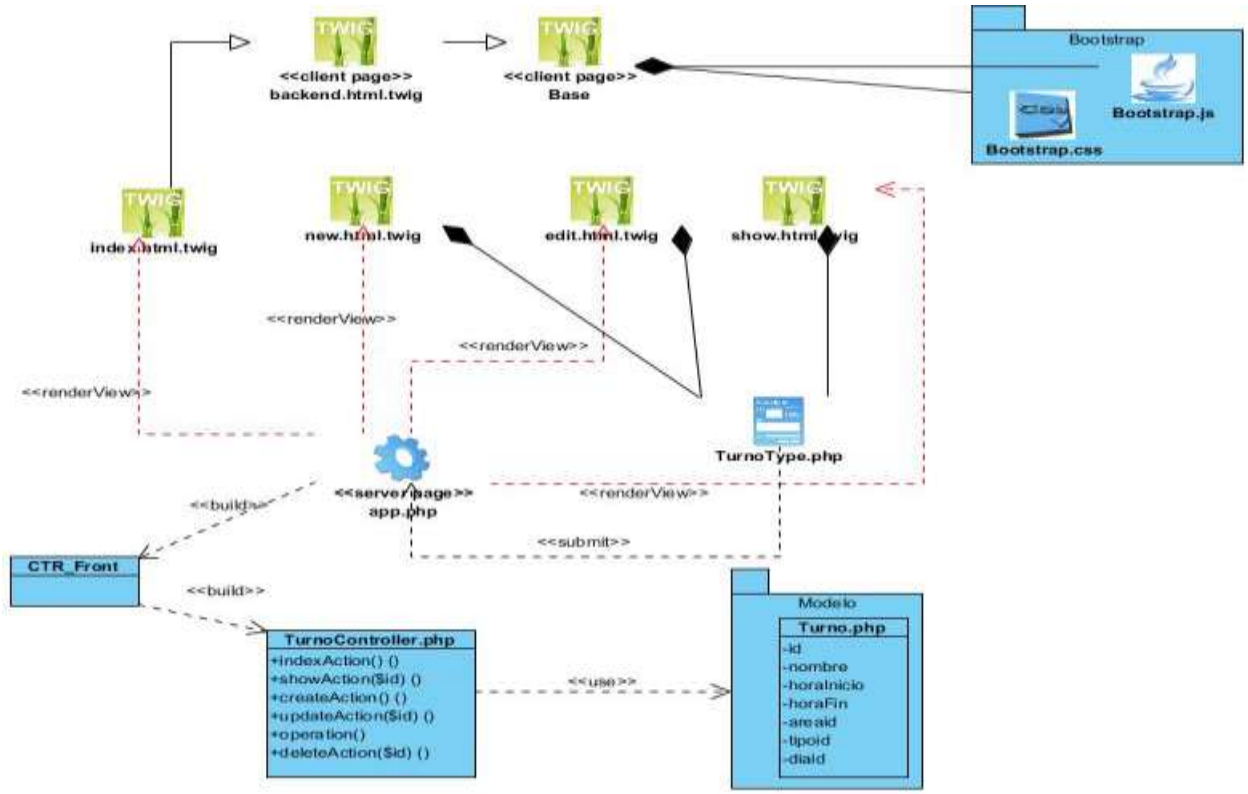
Clase Traza	
Responsabilidad	**Colaboración**
Crear Listar Detallar	Usuario

Anexo 3 Diagramas de clases del diseño con estereotipos web




TWIG
<<client page>>
Base
Bootstrap
Bootstrap.js
Bootstrap.css
<<client page>>
backend.html.twig
<<renderView>>
index.html.twig
new.html.twig
show.html.twig
edit.html.twig
AreaController
+indexAction()
+showAction($id) ()
+newAction()()
+editAction($id) ()
+updateAction()
+deleteAction($id) ()
+createAction() ()
modelo
Area.php
-id
-denominacion
<<use>>
<<link>>
<<renderView>>
<<renderView>>
<<renderView>>
AreaType.php
<<submit>>
<<build>>
CTR_Front
<<server page>>
app.php
<<build>>


TWIG
<<client page>>
Base
Bootstrap
Bootstrap.js
Bootstrap.css
<<client page>>
backend.html.twig
modelo
Posta.php
-id
-nombre
-descripcion
-areaid
-tipoid
edit.html.twig
show.html.twig
new.html.twig
index.html.twig
<<use>>
<<renderView>>
<<renderView>>
<<renderView>>
PostaType.php
<<submit>>
PostaController.php
+indexAction()
+showAction($id)
+newAction()
+createAction()
+editAction($id)
+updateAction($id)
+deleteAction($id)
<<link>>
<<build>>
<<renderView>>
<<server page>>
app.php
<<build>>
CTR_Front



Anexo 4 Ejemplo de patrones

```
 * @ORM\Table(name="Usuario")
 * @ORM\Entity(repositoryClass="GOE\Plan
 */
class Usuario implements UserInterface,
```

Figura 11Ejemplo del patrón GRASP Bajo Acoplamiento

Anexo 5 Tareas detalladas Iteración 1

Tarea de programación	
Número: 3	**Número Historia**: 1
Nombre de la tarea: Escribir código fuente para adicionar o editar un área.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntosestimados**:1
Programador responsable:	
Descripción: Una vez seleccionada la opción de crear un área, se debe mostrar el formulario necesario para la introducción de los datos guardándose en la base de datos, creando así un área.	

Una vez seleccionada la opción de editar un área, se debe mostrar el formulario necesario para cambiar los datos y actualizarlos en la base de datos y la opción de eliminar el área.

Tabla 56Descripción de la tarea de programación #3

Tarea de programación	
Número: 4	**Número Historia**: 2
Nombre de la tarea: Crear un menú para la administración de configuraciones de área.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Crear un menú donde se pueda acceder a las configuraciones de un área que son administración de usuarios, postas, turnos y roles. Donde se muestre una interfaz con los elementos que propone el administrador de área.	

Tabla 57Descripción de la tarea de programación #4

Tarea de programación	
Número: 5	**Número Historia**: 2
Nombre de la tarea: Crear una interfaz para cada configuración del área.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Crear una interfaz para cada configuración donde se muestre una lista con todos sus elementos y se puedan adicionar o modificar.	

Tabla 58Descripción de la tarea de programación #5

Tarea de programación	
Número:6	**Número Historia**: 2
Nombre de la tarea: Listar elementos de las configuraciones.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntosestimados**:0.5
Programador responsable:	
Descripción:	

Listar los elementos de la configuración seleccionada.

Tabla 59Descripción de la tarea de programación #6

Tarea de programación	
Número: 7	**Número Historia**: 2
Nombre de la tarea: Escribir código fuente para adicionar o modificarlas configuraciones del área.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 2
Programador responsable:	
Descripción: Crear la opción de adicionar las configuraciones antes mencionadas, donde se debe mostrar el formulario correspondiente para introducir los datos. Crear la opción de modificar las configuraciones, donde se debe mostrar el formulario necesario para cambiar los datos y actualizarlos en la base de datos y la opción de eliminar.	

Tabla 60Descripción de la tarea de programación #7

Tarea de programación	
Número: 8	**Número Historia**: 3
Nombre de la tarea: Crear interfaz para planificar guardia.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntosestimados**:2
Programador responsable:	
Descripción: Crear una interfaz con los elementos que propone el administrador de área. Donde se pueda escoger un intervalo de tiempo para realizar la planificación y se pueda definir cuantas personas se planificaran por postas. También es necesario que se pueda visualizar el resultado de la planificación.	

Tabla 61Descripción de la tarea de programación #8

Tarea de programación	
Número: 9	**NúmeroHistoria**:1
Nombre de la tarea: Escoger un rango de fecha a planificar.	

Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 2
Programador responsable:	
Descripción: En esta tarea se debe implementar el código que permita al usuario escoger un rango de fecha para generar una planificación.	

Tabla 62Descripción de la tarea de programación #9

Tarea de programación	
Número: 10	**NúmeroHistoria**:1
Nombre de la tarea: Escribir código fuente para procesar los datos y planificar guardia a estudiantes	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 4
Programador responsable:	
Descripción: Escribir un código que tome datos tales como usuarios, postas y turnos, los distribuya y se obtengan las guardias asociadas a esa planificación.	

Tabla 63Descripción de la tarea de programación #10

Anexo 6 Tareas detalladas iteración 2

Tarea de programación	
Número: 13	**Número Historia**:4
Nombre de la tarea: Escribir código fuente para procesar los datos y planificar guardia a trabajadores	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 4
Programador responsable:	
Descripción: Escribir un código que tome datos tales como usuarios, postas y turnos, los distribuya y se obtengan las guardias asociadas a esa planificación.	

Tabla 64Descripción de la tarea de programación #13

Tarea de programación

Número: 14	**Número Historia**: 5
Nombre de la tarea: Crear opción para planificar a los controladores.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**:1
Programador responsable:	
Descripción: Una vez realizada una de las planificaciones antes realizadas su interfaz debe mostrar una opción para distribuir los controladores a las guardias asociadas a esa planificación.	

Tabla 65Descripción de la tarea de programación #14

Tarea de programación	
Número: 15	**NúmeroHistoria**:5
Nombre de la tarea: Escribir código fuente para procesar los datos y planificar guardia a planificadores	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Escribir un código que distribuya a los controladores por las guardias existentes en la planificación.	

Tabla 66Tabla 67Descripción de la tarea de programación #15

Anexo 7 Tareas detalladas iteración 3

Tarea de programación	
Número: 19	**Número Historia**: 6
Nombre de la tarea: Listar guardias asociadas a la publicación.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**:0.5
Programador responsable:	
Descripción: Se debe obtener una lista con todas las guardias asociadas a esa publicación.	

Tabla 68Descripción de la tarea de programación #19

Tarea de programación	
Número: 20	**Número Historia**: 7
Nombre de la tarea: Crear interfaz para la evaluar guardia.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**:0.5
Programador responsable:	
Descripción: Crear una interfaz con los elementos que propone el administrador de área. Donde se muestre una lista con las guardias existentes en su área, se puedan filtrar las mismas según una fecha elegida y se pueda modificar las evaluaciones de los usuarios así como evaluar a los planificadores.	

Tabla 69Descripción de la tarea de programación #20

Tarea de programación	
Número: 21	**Número Historia**: 7
Nombre de la tarea: Listar guardias	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Se debe obtener una lista con todas las guardias.	

Tabla 70Descripción de la tarea de programación #21

Tarea de programación	
Número: 22	**NúmeroHistoria**:7
Nombre de la tarea: Escoger una fecha.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: En esta tarea se debe implementar el código que permita al usuario escoger una fecha para filtrar las guardias realizadas en la misma.	

Tabla 71Descripción de la tarea de programación #22

Tarea de programación

Número: 23	**NúmeroHistoria**:7
Nombre de la tarea: Escribir código fuente para que se pueda emitir una evaluación.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: En esta tarea se debe implementar el código que permita al Administrador de área evaluar o modificar la evaluación de un usuario.	

Tabla 72Descripción de la tarea de programación #23

Tarea de programación	
Número: 24	**NúmeroHistoria**:8
Nombre de la tarea: Crear una interfaz para poder emitir una evaluación.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Crear una interfaz con los elementos que propone el Controlador. Donde se muestre una lista con las guardias que aún no ha evaluado y pueda emitir una evaluación.	

Tabla 73Descripción de la tarea de programación #24

Tarea de programación	
Número: 25	**NúmeroHistoria**:8
Nombre de la tarea: Listar usuarios no evaluados.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: En esta tarea se debe implementar el código que permita obtener una lista de usuarios no evaluados relacionados con el controlador autenticado.	

Tabla 74Descripción de la tarea de programación #25

Tarea de programación	
Número: 26	**NúmeroHistoria**:8
Nombre de la tarea: Escribir código fuente para que se pueda emitir una evaluación.	

Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: En esta tarea se debe implementar el código que permita al Administrador de área evaluar o modificar la evaluación de un usuario.	

Tabla 75Descripción de la tarea de programación #26

Tarea de programación	
Número: 27	**NúmeroHistoria**:9
Nombre de la tarea: Crear una interfaz para reportar incidencia.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Crear una interfaz con los elementos que son propuestos para reportar incidencia.	

Tabla 76Descripción de la tarea de programación #27

Tarea de programación	
Número: 28	**NúmeroHistoria**:9
Nombre de la tarea: Escribir código fuente para reportar incidencia	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Una vez en la interfaz de reportar incidencia, se debe mostrar el formulario necesario para la introducción de los datos guardándose en la base de datos, creando así una incidencia.	

Tabla 77Descripción de la tarea de programación #28

Tarea de programación	
Número: 29	**NúmeroHistoria**:10
Nombre de la tarea: Crear una interfaz para poder administrar incidencia.	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1

(Desarrollo/Corrección/Mejora)	
Programador responsable:	
Descripción: Crear una interfaz con los elementos que son propuestos para la administración de incidencias. Donde se muestre una lista con las incidencias existentes y se pueda adicionar una incidencia, ver sus detalles o dar una respuesta.	

Tabla 78Descripción de la tarea de programación #29

Tarea de programación	
Número: 30	**NúmeroHistoria**:10
Nombre de la tarea: Listar incidencias	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Listar todas las incidencias que existen en la base de datos	

Tabla 79Descripción de la tarea de programación #30

Tarea de programación	
Número: 31	**NúmeroHistoria**:10
Nombre de la tarea: Escribir código fuente para adicionar, ver detalles o dar respuesta a una incidencia.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Una vez seleccionada la opción de crear una incidencia, se debe mostrar el formulario necesario para la introducción de los datos guardándose en la base de datos. En caso de seleccionar ver detalles se den visualizar los todos los datos de la incidencia y en caso de responder se debe mostrar un formulario donde introduce la respuesta a la incidencia.	

Tabla 80Descripción de la tarea de programación #31

Tarea de programación	
Número: 32	**NúmeroHistoria**:11
Nombre de la tarea: Escribir código fuente para denunciar usuarios a comisión o demerito.	

Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 2
Programador responsable:	
Descripción: Escribir código para que una vez que se introducen las evaluaciones de las guardias, la aplicación debe ser capaz de que todos los usuarios que obtenga evaluación de M por segunda vez sea reportado según el tipo de usuario.	

Tabla 81Descripción de la tarea de programación #32

Tarea de programación	
Número: 33	**NúmeroHistoria**:11
Nombre de la tarea: Crear una interfaz para visualizar las denuncias.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Crear una interfaz con los elementos que son propuestos por el administrador de área para visualizar todas las denuncias en el área.	

Tabla 82Descripción de la tarea de programación #33

Anexo 8 Tareas detalladas iteración 4

Tarea de programación	
Número: 36	**NúmeroHistoria**:12
Nombre de la tarea: Crear código fuente para añadir un usuario	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Escribir el código necesario para añadir un usuario a la lista de usuarios que desean cambiar guardia.	

Tabla 83Descripción de la tarea de programación #36

Tarea de programación	
Número: 37	**NúmeroHistoria**:12

Nombre de la tarea: Crear código fuente para solicitar cambio	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Escribir el código necesario para enviar una solicitud de cambio de guardia.	

Tabla 84Descripción de la tarea de programación #37

Tarea de programación	
Número: 38	**NúmeroHistoria**:12
Nombre de la tarea: Crear interfaz para visualizar las solicitudes	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Crear una interfaz que le muestre al usuario una lista con las personas que solicitan cambiar la guardia con él, donde tenga la opción de aceptarla o denegarla.	

Tabla 85Descripción de la tarea de programación #38

Tarea de programación	
Número: 39	**NúmeroHistoria**:12
Nombre de la tarea: Crear código fuente aceptar o denegar solicitud	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 0.5
Programador responsable:	
Descripción: Escribir el código necesario para la opción aceptar una solicitud y realizar un cambio de guardia o denegar solicitud.	

Tabla 86Descripción de la tarea de programación #39

Tarea de programación	
Número: 40	**Número Historia**:13
Nombre de la tarea: Crear interfaz para la administración de cambio de guardia	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	

Descripción: Crear una interfaz que muestre todas la guardias del área y permita al administrador realizar cambios de guardia.

Tabla 87Descripción de la tarea de programación #40

Tarea de programación	
Número: 41	**Número Historia**:13
Nombre de la tarea: Generar código fuente para cambiar guardia	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Crear el código que permita al administrador de área seleccionar un usuario y cambiar su guardia con otro.	

Tabla 88Descripción de la tarea de programación #41

Tarea de programación	
Número: 42	**Número Historia**:14
Nombre de la tarea: Crear interfaz para visualizar evaluaciones	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Crear una interfaz que le permita al usuario visualizar las evaluaciones obtenidas en la realización de la guardia.	

Tabla 89Descripción de la tarea de programación #42

Tarea de programación	
Número: 43	**Número Historia**:14
Nombre de la tarea: Listar evaluaciones	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Crear código fuente para listar todas las evaluaciones de un usuario.	

Tabla 90Descripción de la tarea de programación #43

Tarea de programación	
Número: 44	**Número Historia**:15
Nombre de la tarea: Crear opciones de cambio	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Una vez realizada una planificación se debe mostrar las guardias y con ellas la opción de seleccionarla y cambiar.	

Tabla 91Descripción de la tarea de programación #44

Tarea de programación	
Número: 45	**Número Historia**:15
Nombre de la tarea: Crear código fuente para ajustar la guardia	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Se crea el código necesario para cambiar las guardias según desee el administrador de área.	

Tabla 92Descripción de la tarea de programación #45

Tarea de programación	
Número: 46	**Número Historia**:16
Nombre de la tarea: Crear interfaz para la generación de reportes	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Se crea una interfaz que muestre los reportes que son generados	

Tabla 93Descripción de la tarea de programación #46

Tarea de programación	
Número: 47	**Número Historia**:16
Nombre de la tarea: Crear código fuente para obtener los reportes	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1

(Desarrollo/Corrección/Mejora)	
Programador responsable:	
Descripción: Crear código necesario para obtener los reportes definidos	

Tabla 94Descripción de la tarea de programación #47

Tarea de programación	
Número: 48	**Número Historia**:16
Nombre de la tarea: Crear interfaz para la generación de estadísticas	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Se crea una interfaz que muestre los reportes que pueden ser generados	

Tabla 95Descripción de la tarea de programación #48

Tarea de programación	
Número: 49	**Número Historia**:16
Nombre de la tarea: Crear código fuente para obtener estadísticas	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos estimados**: 1
Programador responsable:	
Descripción: Crear código necesario para obtener las estadísticas definidas	

Tabla 96Descripción de la tarea de programación #49

Anexo 9 Casos de Prueba

Caso de Prueba de Aceptación	
Código: HU1_P1	Historia de Usuario: Administración de área
Nombre: Gestionar área en el sistema.	
Descripción: Probar que se adicionen, modifiquen y eliminen correctamente los datos de un área en el sistema.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración, los datos del área deben de guardarse en la base de datos del sistema. Se debe permitir que un área pueda ser modificada y eliminada en la base de datos utilizando datos válidos.	

Entrada / Pasos de Ejecución: Se intenta adicionar, modificar y eliminar un área utilizando datos válidos.
Resultado Esperado: Los datos del área son adicionados correctamente en la base de datos del sistema. La posta seleccionada es modificada y eliminada correctamente.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 97 Caso de prueba de HU administración de área

Caso de Prueba de Aceptación	
Código: HU2_P1	Historia de Usuario: Administración de configuraciones del área (Gestionar usuario).
Nombre: Gestionar usuario en el sistema.	
Descripción: Probar que se adicionen correctamente los datos de un usuario en el sistema y además, que se permita desactivar un usuario en la base de datos, sin ser eliminado.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración, los datos del usuario deben de guardarse en la base de datos del sistema. Se debe permitir que un usuario pueda ser desactivado en la base de datos utilizando datos válidos, sin ser eliminado de la misma. Cuando un usuario se desactiva no se toma en cuenta en la planificación de la guardia.	
Entrada / Pasos de Ejecución: Se intenta adicionar y desactivar un usuario utilizando datos válidos.	
Resultado Esperado: Los datos del usuario son adicionados correctamente en la base de datos del sistema. El usuario seleccionado se desactiva en la base de datos sin ser eliminado, quedando excluido de la siguiente planificación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 98: Caso de prueba HU: Administración de configuraciones del área (Gestionar usuario).

Caso de Prueba de Aceptación	
Código: HU2_P2	Historia de Usuario: Administración de configuraciones del área (Gestionar posta).
Nombre: Gestionar posta en el sistema.	
Descripción: Probar que se adicionen, modifiquen y eliminen correctamente los datos de una posta en el sistema.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración, los datos de la posta deben de guardarse en	

la base de datos del sistema. Se debe permitir que una posta pueda ser modificada y eliminada en la base de datos utilizando datos válidos.
Entrada / Pasos de Ejecución: Se intenta adicionar, modificar y eliminar una posta utilizando datos válidos.
Resultado Esperado: Los datos de la posta son adicionados correctamente en la base de datos del sistema. La posta seleccionada es modificada y eliminada correctamente.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 99Caso de prueba HU: Administración de configuraciones del área (Gestionar posta).

Caso de Prueba de Aceptación	
Código: HU2_P3	Historia de Usuario: Administración de configuraciones del área (Gestionar turno).
Nombre: Gestionar turno en el sistema.	
Descripción: Probar que se adicionen, modifiquen y eliminen correctamente los datos de un turno en el sistema.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración, los datos del turno deben de guardarse en la base de datos del sistema. Se debe permitir que un turno pueda ser modificado y eliminado en la base de datos utilizando datos válidos.	
Entrada / Pasos de Ejecución: Se intenta adicionar, modificar y eliminar un turno utilizando datos válidos.	
Resultado Esperado: Los datos del turno son adicionados correctamente en la base de datos del sistema. La posta seleccionada es modificada y eliminada correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 100Caso de prueba HU: Administración de configuraciones del área (Gestionar turno).

Caso de Prueba de Aceptación	
Código: HU3_P1	Historia de Usuario: Planificar guardia a estudiantes
Nombre: Planificar Estudiantes.	
Descripción: Probar que se realice la planificación de los usuarios de tipo estudiantes de forma correcta.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración. Se debe permitir la selección de un rango de tiempo a planificar.	
Entrada / Pasos de Ejecución: Se selecciona la fecha de inicio y fin. Se	

genera una distribución, la cual puede ser ajustada. Se asignan los controladores y se publica la planificación.
Resultado Esperado: Los usuarios de tipo estudiante son distribuidos en postas y turnos de guardia dentro del rango de fechas seleccionado.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 101Caso de prueba HU: Planificar guardia a estudiantes

Caso de Prueba de Aceptación	
Código: HU4_P1	Historia de Usuario: Planificar guardia a trabajadores
Nombre: Planificar Trabajadores.	
Descripción: Probar que se realice la planificación de los usuarios de tipo trabajador de forma correcta.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración. Se debe permitir la selección de un rango de tiempo a planificar.	
Entrada / Pasos de Ejecución: Se selecciona la fecha de inicio y fin. Se genera una distribución, la cual puede ser ajustada. Se asignan los controladores y se publica la planificación.	
Resultado Esperado: Los usuarios de tipo trabajador son distribuidos en postas y turnos de guardia dentro del rango de fechas seleccionado.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 102Caso de prueba HU: Planificar guardia a trabajadores

Caso de Prueba de Aceptación	
Código: HU5_P1	Historia de Usuario: Planificar guardia a controladores
Nombre: Distribuir Controladores.	
Descripción: Probar que se realice la distribución de los usuarios con el rol controlador de forma correcta.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración. Se debe de tener guardias sin controladores asignados.	
Entrada / Pasos de Ejecución: Se presiona un botón para comenzar la asignación de controladores.	
Resultado Esperado: Los usuarios activos con el Rol controlador, son asignados a las guardias creadas que no poseen controladores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 103Caso de prueba HU: Planificar guardia a controladores

Caso de Prueba de Aceptación	
Código: HU5_P1	Historia de Usuario: Planificar guardia a controladores
Nombre: Distribuir Controladores.	
Descripción: Probar que se realice la distribución de los usuarios con el rol controlador de forma correcta.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración. Se debe de tener guardias sin controladores asignados.	
Entrada / Pasos de Ejecución: Se presiona un botón para comenzar la asignación de controladores.	
Resultado Esperado: Los usuarios activos con el Rol controlador, son asignados a las guardias creadas que no poseen controladores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 104Caso de prueba HU: Planificar guardia a controladores

Caso de Prueba de Aceptación	
Código: HU7_P1	Historia de Usuario: Evaluar guardia
Nombre: Evaluar guardia.	
Descripción: Probar que se emita la evaluación de la guardia de un usuario.	
Condiciones de Ejecución: La aplicación debe ejecutarse con privilegios de administración. Se debe de tener guardias planificadas.	
Entrada / Pasos de Ejecución: Se selecciona una fecha y se listan las guardias planificadas en la misma. Se selecciona una evaluación y se asigna a una guardia.	
Resultado Esperado: Se emite una evaluación de una guardia determinada.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 105Caso de prueba HU: Evaluar guardia

Caso de Prueba de Aceptación	
Código: HU7_P1	Historia de Usuario: Emitir evaluación
Nombre: Emitir evaluación.	
Descripción: Probar que se emita la evaluación de la guardia de un	

usuario.
Condiciones de Ejecución: La aplicación debe ejecutarse con privilegios de controlador. Se debe de tener guardias planificadas.
Entrada / Pasos de Ejecución: Se selecciona una guardia, y se le asigna una evaluación.
Resultado Esperado: Se emite una evaluación de una guardia determinada.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 106Caso de prueba HU: Emitir evaluación

Caso de Prueba de Aceptación	
Código: HU9_P1	Historia de Usuario: Reportar incidencias
Nombre: Reportar incidencias.	
Descripción: Probar que se puede reportar incidencias.	
Condiciones de Ejecución: Los usuarios deben de estar autenticado. Los usuarios deben de tener el rol de controlador.	
Entrada / Pasos de Ejecución: Se intenta adicionar una incidencia.	
Resultado Esperado: Se crea una nueva incidencia y se almacena en la base de dato satisfactoriamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 107Caso de prueba HU: Reportar incidencias

Caso de Prueba de Aceptación	
Código: HU10_P1	Historia de Usuario: Administrar incidencias
Nombre: Responder y listar incidencias.	
Descripción: Probar que se listen incidencias y se dé respuestas a las mismas.	
Condiciones de Ejecución: La aplicación debe estar ejecutado con privilegios de administración. Se debe tener incidencias reportadas para responder.	
Entrada / Pasos de Ejecución: Se listan las incidencias reportadas. Se da respuesta a la incidencia. Se elimina correctamente la incidencia.	
Resultado Esperado: Se listan, responden y eliminan las incidencias correctamente de la base de datos.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 108Caso de prueba HU: Administrar incidencias

Caso de Prueba de Aceptación	
Código: HU11_P1	Historia de Usuario: Reportar a comisión disciplinaria y demerito
Nombre: Generar reporte de comisión disciplinaria.	
Descripción: Probar que se genere un reporte de los usuarios que deben ser denunciados a comisión disciplinaria o demeritados.	
Condiciones de Ejecución: La aplicación debe ejecutarse con privilegios de administración. Deben de existir usuarios evaluados de Mal dos veces consecutivas.	
Entrada / Pasos de Ejecución: Se evalúa a un usuario de mal dos veces consecutivas y se intenta visualizar el reporte de los usuarios que deben ser denunciados a comisión disciplinaria o demeritados.	
Resultado Esperado: Se genera un reporte de usuarios que deben ser denunciados a comisión disciplinaria o demeritados.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 109Caso de prueba HU: Reportar a comisión disciplinaria y demerito

Caso de Prueba de Aceptación	
Código: HU12_P1	Historia de Usuario: Solicitud y cabio de guardia
Nombre: Cambiar Guardia.	
Descripción: Probar que se cambien de guardia dos usuarios.	
Condiciones de Ejecución: Los Usuarios deben estar autenticados. Los usuarios deben de tener los privilegios necesarios para realizar esta acción. Se deben de tener guardia planificada.	
Entrada / Pasos de Ejecución: Se envía una solicitud de cambio de guardia a un usuario. Se acepta la solicitud de cambio de guardia.	
Resultado Esperado: Los usuarios son intercambiados en las guardias asociadas a los mismos.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 110Caso de prueba HU: Solicitud y cabio de guardia

Caso de Prueba de Aceptación	
Código: HU14_P1	Historia de Usuario: Visualizar evaluaciones
Nombre: Ver evaluaciones de la guardia.	
Descripción: Probar que se genere correctamente un reporte de las evaluaciones del usuario.	

Condiciones de Ejecución: Los Usuarios deben estar autenticados. Se debe de tener guardia planificada.
Entrada / Pasos de Ejecución: Se intenta generar el reporte de las evaluaciones del usuario.
Resultado Esperado: Se genera un reporte de las evaluaciones del usuario.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 111Caso de prueba HU: Visualizar evaluaciones

Caso de Prueba de Aceptación	
Código: HU15_P1	Historia de Usuario: Ajuste de planificación
Nombre: Ajustar planificación.	
Descripción: Probar que puedan realizar cambios de guardia antes de asignar los controladores y hacer pública la planificación.	
Condiciones de Ejecución: La aplicación debe ejecutarse con privilegios de administración. Debe de haberse generado una planificación no publicada.	
Entrada / Pasos de Ejecución: Se selecciona un usuario perteneciente a una guardia. Se cambia con otro usuario de otra guardia.	
Resultado Esperado: Se realiza el ajuste de los usuarios antes de publicar la planificación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 112Caso de prueba HU: Ajuste de planificación

Caso de Prueba de Aceptación	
Código: HU16_P1	Historia de Usuario: Emitir reporte
Nombre: Reportes por Área.	
Descripción: Probar que se generen correctamente los reportes pertenecientes a un área determinada.	
Condiciones de Ejecución: La aplicación debe ser ejecutada con privilegios de administración. Los datos de los reportes del área deben de visualizarse correctamente.	
Entrada / Pasos de Ejecución: Se intenta generar los reportes de un área.	
Resultado Esperado: Los reportes del área se generan correctamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 113 Caso de prueba HU: Emitir reporte

Caso de Prueba de Aceptación	
Código: HU17_P1	Historia de Usuario: Visualizar estadísticas
Nombre: Mostrar estadísticas.	
Descripción: Probar que visualicen datos estadísticos definidos mediante gráficas.	
Condiciones de Ejecución: La aplicación debe ejecutarse con privilegios de administración. Deben de existir datos estadísticos.	
Entrada / Pasos de Ejecución: Se intenta generar estadísticas a partir de la base de datos.	
Resultado Esperado: se muestra un gráfico con estadística obtenida de la base de datos.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 114 Caso de prueba HU: Visualizar estadísticas