UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

Sistema de planificación y control de la Guardia Obrera Estudiantil

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR

José Angel Díaz Romero.

Tutores: Lic. Antonio Gutiérrez Laborit.

Ing. Risell Ramírez Ramos.

La Habana, 2014

"Año 56 de la Revolución"

DECLARACIÓN DE AUTORÍA

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

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

José Angel Díaz Romero

_____

Autor

Antonio Gutiérrez Laborit

_____

Tutor

Risell Ramírez Ramos

_____

Tutora

"… La primera obligación de cada ser humano es ser feliz, la segunda es hacer feliz a los demás…"
Mario Moreno (Cantinflas)

AGRADECIMIENTOS

Este trabajo de diploma es el resultado del esfuerzo conjunto de varias personas que me apoyaron y aconsejaron cuando más lo necesité, no solo durante la carrera sino durante toda mi vida. Es por ello que quiero agradecerles a las siguientes personas:

A mi mamá por ser la mujer que me dio la vida y me enseñó, por darme su cariño y amor desde que nací, sin ti seguro mi vida no sería tan maravillosa como lo es hoy, te debo todo lo que tengo y todo lo que soy. Nunca podré agradecerte tantas cosas que has hecho por mí. Eres la persona más importante en mi vida y quiero que lo recuerdes siempre.

A mi hermanita Rosalia Daimé Díaz Romero por apoyarme y darme su aliento cada día. Por compartir conmigo momentos alegres y tristes. Estoy muy orgulloso de poder ser tu hermana mayor, tu protector. Aunque no te lo diga con frecuencia sabes que te quiero y te quiero muchísimo. Estoy muy orgulloso con la hermanita que mami y papi me dieron.

A mi padre, te quiero mucho, mucho. Muchas gracias, por darme cariño de veras papá. Muy orgulloso estoy de ti.

A mi hermano Aciel Gonzales Espinosa, gracias por estar siempre conmigo, de corazón te lo digo te quiero mucho, mi amigo del alma.

A mi abuelo: Erain y a mis abuelas Camelo y Josefa; a mis tíos: Idalberto, Onar, Yoel; a mis tías Sonia, Licet, Alina; a mi primita Yeni; a mis primos Ivancito, José Antonio y Yunier; a la meyi mi vecina gracias por todo el apoyo que me brindaron.

A todos aquéllos que contribuyeron en mi formación académica y profesional: a mis profesores, que compartieron conmigo sus conocimientos a lo largo de mi educación universitaria; especialmente a mi tutor Toni, gracias hermano por apoyarme y mi tutora Risell, gracias profesora por ser paciente conmigo y apoyarme.

A mis amigos de la universidad, desde 1ero hasta 5to, principalmente a Alejandro Castellano Loaces y Jeymel Cosme Hernández, gracias por

todo, siempre en mi lecho habrá refugio y gratitud inagotable para con ustedes.

A la gente de la antigua cueva: Yoandris, Yordanis, Noa, Infante, Orelbis, Omar, Yobannis, Edelso, Marco.

A la gente de la nueva cueva 135-308: El Fliper, Tito, El Lachi, el jefe Crod Leo, Maikel y Carlos,

A los colegas del futbol 11 Vilson, Yusniel, Eduardo, Marlon.

En fin a todas las personas que formaron y forman parte de mi vida, gracias.

DEDICATORIA

Dedico este trabajo de diploma a mi abuelito Ramonín y a mi mamá Chela, aún los amo con la vida y nunca he de olvidarlos.

También quiero dedicar este resultado a quien a lo largo de mi vida ha estado a mi lado, apoyándome y siempre intentando guiarme por buenos senderos. Por ser fuerte y ser siempre como eres conmigo: dura en ocasiones y recia en las otras; aprendí de ti a ser fuerte, a siempre dar la mano a quien la necesita y entre otras cosas a vivir de forma honrada. Te amo mucho mamá, con toda mi alma.

RESUMEN

El desarrollo científico y tecnológico alcanzado por la humanidad en la rama de la informática y las comunicaciones, posibilita agilizar los métodos de gestión de información, enseñanza y aprendizaje. La Universidad de las Ciencias Informáticas comprende entre sus objetivos la construcción de software, varios de los procesos que se realizan han sido migrados hacia una aplicación que permita lograr un mayor control y disminuya la ocurrencia de errores durante la realización de los mismos, ejemplo de esto, es la guardia obrero-estudiantil que se planifica y gestiona de forma manual mediante la herramienta Hojas de Cálculos (*Microsoft Excel*) correspondiente al Paquete de Oficina (*Microsoft Office*).

En tal sentido, el presente trabajo de diploma tiene como objetivo desarrollar una aplicación que permita gestionar y planificar la guardia obrero-estudiantil en la Facultad 4 perteneciente a la Universidad de las Ciencias Informáticas. Recoge los resultados de la investigación realizada, describiéndose las principales características de los sistemas homólogos analizados, así como la arquitectura y las herramientas utilizadas para la implementación de la propuesta de solución. Se describen las tecnologías empleadas y los artefactos generados en el proceso de desarrollo. Como resultado se obtiene un software que permite optimizar el proceso de planificación y gestión de la guardia obrero-estudiantil, el cual podrá ser extendido a otros centros que contemplen este tipo de actividad.

Palabras claves: brigada, control, gestión, planificación, servicio de guardia, turno, zona.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

La expansión de las Tecnologías de la Información y las Comunicaciones (TIC) sobre las diversas esferas de la sociedad ha representado un indudable cambio de paradigma en la forma de accionar de los hombres. Los novedosos avances de la informática como ciencia, han influido de manera positiva en la informatización de procesos, proporcionando a la sociedad novedosas herramientas que faciliten la realización de las actividades diarias. En Cuba, un país subdesarrollado del tercer mundo este tema tiene sus peculiaridades, fomentando su uso en esferas que son priorizadas por el estado, tales como la educación, la salud y el deporte.

La gestión de los procesos del Ministerio de la Educación Superior no ha escapado a esta realidad que sin lugar a dudas ha transformado las formas de planificar, organizar, ejecutar y controlar determinados flujos de trabajo que antes se realizaban de forma rudimentaria.

El Servicio de Guardia Obrero-Estudiantil (SGOE) es uno de los procesos sustantivos que se realizan en la Universidad de las Ciencias Informáticas (UCI). El mismo es de vital importancia, no solo porque en él se confía la seguridad de los medios materiales, personales y estatales, así como la integridad física de los ciudadanos que residen en ella, sino por el impacto educativo que este proceso tiene para profesores, trabajadores y estudiantes.

En la UCI existen diferentes facultades que llevan a cabo el proceso de guardia obrero-estudiantil, específicamente en la Facultad 4 la planificación del servicio de guardia se realiza manualmente haciendo uso de la herramienta *Microsoft Excel*, lo cual lo convierte en un proceso lento, tedioso, agotador y propenso a errores de planificación como: repetición de turnos y zonas, no tener en cuenta la disponibilidad de los recursos humanos, no tener en cuenta si es residente o no en la universidad y ubicar féminas fuera del primer y segundo turno del SGOE.

La Dirección de Seguridad y Protección, encargada de velar por el cumplimiento del servicio de guardia, le exige a la Facultad enviar partes diarios y mensuales sobre el cumplimiento de la guardia, lo cual genera una gran cantidad de información no digitalizada, gastos significativos en recursos materiales y derroche de tiempo de trabajo del personal encargado de realizar esta labor.

De igual forma para hacer efectivo el proceso de planificación y control del SGOE, es imprescindible difundir la planificación con antelación (mayor a un mes), debido a que existen factores que son importantes a tener en cuenta como: la posibilidad de efectuar intercambios de turnos entre pares de usuarios teniendo en cuenta su categoría, la necesidad de realizar reajustes en la agenda personal o

plan de trabajo de las personas y el hecho de realizar la distribución de la planificación vía correo electrónico.

A pesar de que en la Facultad 4 existen los medios informáticos, la infraestructura y el personal calificado para desarrollar un sistema que permita un control más centralizado de dicho proceso, se efectúan de manera manual los procesos involucrados en la planificación y control del SGOE, enviándose por correo electrónico la hoja de cálculo con la planificación, lo que impide que dos usuarios puedan editar el documento a la vez desde sus puestos de trabajo.

Teniendo en cuenta la situación anteriormente descrita surge como **problema de investigación**: ¿Cómo hacer más eficiente el proceso de planificación y control del SGOE en la Facultad 4 de la UCI?

De lo planteado anteriormente se deriva como **objeto de estudio** la gestión de procesos universitarios.

Determinando para la investigación como **campo de acción** el proceso de desarrollo de sistemas informáticos para la planificación y control del SGOE en la facultad 4 de la UCI.

Para dar solución al problema antes expuesto se formula el siguiente **objetivo general:** Desarrollar un sistema informático para hacer más eficiente el proceso de planificación y control del SGOE en la Facultad 4 de la UCI.

Para dar cumplimiento al objetivo general, se definen como **objetivos específicos:**

- Elaborar los referentes teóricos-metodológicos fundamentales relacionados con la gestión de procesos universitarios en general y con la planificación y control de servicios de guardia mediante aplicaciones informáticas.
- Identificar los requerimientos con que debe cumplir la propuesta de solución.
- Implementar el sistema de planificación y control del SGOE.
- Realizar pruebas de *software* a la propuesta de solución.

Proponiendo como **idea a defender** que con el desarrollo de un sistema informático se hace más eficiente el proceso de planificación y control del SGOE en la Facultad 4 de la UCI.

Las tareas de la investigación a realizar para dar solución a los objetivos específicos son:

- Revisión bibliográfica para conformar el estado del arte de la investigación mediante la búsqueda de información sobre soluciones similares aplicadas en diversos entornos.
- Diagnóstico del proceso actual de planificación y control del SGOE en la Facultad 4 de la UCI.

- Definición de los requerimientos que debe cumplir la propuesta de solución a desarrollar.
- Selección de la metodología, herramientas y tecnologías para guiar el proceso de desarrollo e implementar el sistema informático.
- Confección de los artefactos que propone la metodología de desarrollo de *software* seleccionada e implementación de la propuesta de solución.
- Realización de las pruebas de software a la propuesta de solución.

Los **métodos científicos** utilizados en la investigación estuvieron determinados por el objetivo general y las tareas de investigación previstas. A nivel teórico fue utilizado el método: **analítico – sintético**, para realizar un estudio bibliográfico profundo de la teoría existente alrededor del objeto de estudio, determinar las características que tendrá la propuesta de solución, y definir las tecnologías y herramientas más adecuadas para el desarrollo de la propuesta de solución. También fue utilizado a nivel empírico el método **entrevista**, para identificar las necesidades reales de las personas involucradas con el SGOE en la Facultad 4 de la UCI y poder obtener las funcionalidades con que contará el sistema.

Para una mejor comprensión de la investigación, cuyo diseño metodológico se acaba de describir, se decidió definir una **estructura capitular** que aporte cierto grado de organización y facilite el estudio del documento. Los capítulos que lo conforman, son los siguientes:

Capítulo 1. Fundamentación teórica: Se analizan los conceptos fundamentales acerca de la gestión y planificación de la guardia obrero-estudiantil (turno, zona, brigada y otros) y simultáneamente se tratan aspectos elementales relacionados con las tecnologías a emplear en el desarrollo de la aplicación. Se realiza un estudio detallado de la metodología que guiará el proceso de desarrollo de *software*.

Capítulo 2. Exploración y planificación: Se identifican los conceptos asociados al dominio del problema y los requisitos funcionales y no funcionales de la solución.

Capítulo 3. Implementación y prueba: Se describen los componentes asociados a la aplicación, se definen los estándares de codificación utilizados durante el desarrollo y se diseña una serie de casos de pruebas con el propósito de validar la aplicación, la evaluación de su ejecución y se describen los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se realizará un estudio de la situación actual del proceso de extensión universitaria servicio de guardia obrero-estudiantil en la UCI, así como la definición de conceptos asociados al mismo. Se describen además las tecnologías actuales de desarrollo utilizadas para el análisis, diseño e implementación del sistema sobre las cuales se apoya la propuesta de solución a desarrollar.

1.1 Conceptos generales relacionados con la investigación

Para un buen entendimiento de la investigación que se desglosará en el presente capítulo es necesario sentar una posición respecto a conceptos claves que serán usados en la misma, los cuales se muestran a continuación.

Planificar: Plan general, metódicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo determinado.

Turno: Orden según el cual se suceden varias personas en el desempeño de cualquier actividad o función.

Zona: Extensión considerable de terreno o superficie encuadrada entre ciertos límites.

Brigada: Conjunto de personas reunidas para dedicarlas a ciertos trabajos.

Reporte: Descripción, oral o escrita, de las características y circunstancias de un suceso o asunto.

Asistencia: Acción de estar o hallarse presente.

IDE: Software que comprende entre otras funcionalidades, un editor, un compilador y un depurador de código de uno o varios lenguajes de programación, en muchas ocasiones incluye un constructor de interfaces visuales.

Librería: En el marco del presente trabajo, se define como una colección o conjunto de funciones usadas para desarrollar software.

1.2 Gestión de procesos universitarios en la UCI

En la UCI se gestionan tres procesos fundamentales, los cuales son:

1. Docencia.
2. Extensión universitaria.

3. Investigación y postgrados.

Como parte de las actividades que se desarrollan en el proceso extensión universitaria se encuentran: Mi beka más bonita, juegos Mella, festival de artistas aficionados, SGOE, entre otros. Precisamente el SGOE está distribuido en las 7 facultades que posee la Universidad donde se le asignan zonas de vigilancia específicas a cada una.

La Facultad 4 de la UCI cuenta con dos zonas a proteger, las mismas son: Docente y Residencia (ver Ilustración 8). La zona Residencia cuenta con dos turnos del SGOE que cubrirán el horario de 11:00 pm a 7:00 am del día próximo, este comportamiento se mantendrá durante toda la semana (la constituyen la semana hábil y el fin de semana). La zona Docente tiene dos comportamientos distintos uno para la semana hábil (de lunes a viernes) y otro para el fin de semana (sábado y domingo). En la semana hábil la zona cuenta con tres turnos del SGOE, los cuales cubren el horario de 7:00 pm a 7:00 am del día siguiente y el fin de semana cuenta con 5 turnos del SGOE, cubriendo el horario de 7:00 am a 7:00 am del siguiente día (ver Anexo 1).

Para la asignación de los turnos a ocupar durante la prestación del SGOE a cada persona, el encargado de la planificación debe tener en cuenta aspectos que son comunes para profesores y estudiantes, que en ocasiones son difíciles de controlar, como son: ¿Está de vacaciones?, ¿Está disponible?, ¿Qué categoría posee? (categorías: Profesor y estudiante), ¿Es residente en la UCI? (en caso de no ser residente en la universidad se le asigna un local para su alojamiento), ¿Es de sexo femenino? (en caso de serlo se tendrá en cuenta si está embarazada o si tiene hijos y que no se le asigne el último turno del SGOE).

1.3 Estudio de sistemas homólogos

Para realizar la presente investigación se realizó un estudio de soluciones que resuelven un problema similar al planteado en el marco teórico-metodológico. Partiendo de la idea de adquirir y reutilizar funcionalidades que tributen a la gestión y planificación del SGOE son analizados los siguientes sistemas:

TURNEX: Es un producto creado por la empresa *SOLEX* de origen chilena y se diseñó para programar y planificar los turnos de los empleados de forma fácil y eficiente. Su atención se centra en resolver complejas asignaciones y cambios en los respectivos turnos. Está orientado a todas aquellas empresas que prestan servicios de personal a sus clientes y que trabajan en base a turnos, como por ejemplo: guardias de seguridad, empleados, entre otros. (*SOLEX*, 2010)

Características:(*SOLEX*, 2010)

- Permite la reprogramación de turnos de los empleados.
- Posibilita un mejor aprovechamiento de los recursos humanos.
- Permite visualizar la información de la planificación de los turnos de guardias de los empleados.
- Permite el acceso concurrente de usuarios, ya que es un producto 100% web.
- Permite la gestión de zonas.
- Permite controlar la asistencia de los agentes de seguridad.

TURNEX responde en gran medida a las necesidades existentes actualmente en la Facultad 4 en cuanto a la planificación y control de la guardia obrero-estudiantil, sin embargo es un software privativo lo que conllevaría en caso de que se quiera obtener, a un contrato con la empresa propietaria, conduciendo a un gasto monetario considerable.

Gestor Web para el control de la Guardia Obrera de la Universidad de las Ciencias Informáticas: Se desarrolla con el objetivo de mejorar el control y la obtención de reportes referente al proceso de Control de la Guardia Obrera en la UCI. Es independiente del sistema operativo donde se ejecute y presenta un requerimiento de hardware mínimo, brinda una amplia información sobre todo el tema perteneciente al control de la guardia obrera, minimiza el trabajo manual, agiliza el flujo de información y ahorra recursos a la universidad. El sistema cuenta con una interfaz *web* amigable y fácil de usar permitiendo que los usuarios puedan disfrutar de sus servicios.

Funcionalidades que el sistema permite:

- Gestión de las postas[1].
- Gestión de los turnos[2] de guardia.
- Gestión de las áreas.
- Gestión de los trabajadores.
- Planificar la guardia a los trabajadores.

[1] Son las zonas o áreas a ocupar durante la prestación del SGOE, ejemplo: Manzanas 30 y 31 de la UCI.

[2] Son los horarios planificados para cada brigada que compone el SGOE en la Facultad 4 de la UCI.

- Replanificar la guardia.
- Controlar la guardia planificada.
- Generar reportes e imprimir la información gestionada.

A pesar de ser una solución desarrollada en la UCI no se cuenta con el código fuente de la misma. Fue implementada sin el uso de un *framework* de desarrollo lo que deja una brecha abierta a la seguridad del sistema y al uso de malas prácticas en su construcción. Para realizar la planificación de la guardia no tiene en cuenta la categoría estudiante, solo los trabajadores. Es válido aclarar que el proceso del SGOE ha sufrido cambios con respecto al momento en el que fue desarrollada esta solución.

Plataforma GeoQ-Guardian: La plataforma GeoQ-Guardian fue desarrollada por el Centro de Geo-Informática y Señales Digitales (GEYSE) perteneciente a la UCI surge como consecuencia de la necesidad de agilizar y elevar la eficacia del proceso de planificación y control de la guardia en la universidad.

Funcionalidades que el sistema permite:

- Gestión de las postas.
- Gestión de los turnos de guardia.
- Gestión de las zonas.
- Gestión del personal[3] que realiza la guardia.
- Planificar la guardia, se planifica basado en componentes espaciales.
- Modificar la planificación de la guardia.
- Controlar la guardia planificada.
- Generar reportes[4] e imprimir la información gestionada.

GoeQ-Guardian es una aplicación de escritorio y a pesar de ser una solución lo suficientemente abarcadora, su uso en la planificación y control del SGOE en la Facultad 4 de la UCI se ve imposibilitado.

[3] Se refiere a los estudiantes, profesores y demás personas que integran el SGOE.

[4] El reporte cumplimiento es un informe que exhibe la información del SGOE.

Esto es debido a que para planificar el SG en cada área de la universidad la accesibilidad se ve afectada, ya que solo pueden interactuar con la herramienta las personas del área[5] donde está instalada.

Otra desventaja es que el tiempo de respuesta y la eficiencia disminuyen cuando se cuenta con un elevado cúmulo de datos como es el caso. Para ubicar las personas en el SGOE y las zonas definidas previamente por el responsable en cada área de planificar el SGOE en el mapa de la UCI se usan componentes espaciales y el modelo de cuadrícula que es la propiedad para representar cualquier tipo de imagen digital en mallas. El proceso de mantenimiento y mejoras de la herramienta es engorroso, ya que los administradores tendrían que ir a cada estación de trabajo donde se encuentra desplegado y actualizar los cambios.

Una desventaja es que el *software* pierde la propiedad de ser multiplataforma, siendo funcional solo en entorno *Windows*, lo cual es una restricción a tener en cuenta en un ambiente donde el software libre tiene bastante aceptación. Para realizar el intercambio de turnos en el SG no se tiene en cuenta un acuerdo mutuo entre los implicados. La capacidad de autentificación y autorización de usuarios no es posible, ya que la forma de diseño es centrada en un único usuario local.

Después del análisis de los sistemas similares para la planificación del SGOE: TURNEX, GeoQ-Guardian y el Gestor *web* para el control de la guardia obrera de la UCI, los cuales no se ajustan completamente a las características deseadas para resolver el problema existente en la Facultad 4 de la UCI. Lo que refuerza la necesidad de desarrollar un sistema informático.

Los sistemas analizados, a pesar de no resolver el problema actual, aportaron ideas y funcionalidades para desarrollar una aplicación informática que hará más eficiente el proceso de planificación y control del SGOE en la Facultad 4 de la UCI, estas funcionalidades son:

- Gestión de las postas y zonas.
- Planificar la guardia.
- Controlar la guardia planificada.
- Generar reportes de la información gestionada.

1.4 Estudio de las metodologías de desarrollo de software

[5] Un área es una institución, ejemplo: Facultad 4, Facultad 3, y etc.

Una metodología de desarrollo de *software* es el conjunto de prácticas y reglas empleadas para desarrollar *software*, o sea es el entorno que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información. Una determinada metodología no es necesariamente aplicable a todo tipo de proyectos, más bien cada tipo de proyecto tiene una metodología a la que se adapta mejor. El éxito del producto cae en gran medida sobre la metodología que seleccione el equipo que lo desarrolla en busca de maximizar el potencial del equipo, la calidad del software resultante y el cumplimiento del tiempo de entrega establecido. Las metodologías son clasificadas según su enfoque ágil o tradicional. (Calero Solís, 2003)

Las metodologías con enfoque tradicional están basadas en normas provenientes de estándares seguidos por el entorno de desarrollo y cierta resistencia a los cambios, el cliente interactúa con el equipo de desarrollo mediante reuniones y las mismas generan muchos artefactos. Integran un número importante de roles y están determinadas por grandes grupos de desarrollo. La arquitectura del software es esencial y se expresa mediante modelos. (*Ivar Jacobson*, 2000)

Las metodologías ágiles pueden variar en su ejecución y en su énfasis, tiene como característica el desarrollo iterativo, la integración al equipo de desarrollo del cliente y está determinada por grupos pequeños de desarrollo. Desarrollar mediante iteraciones permite al equipo de desarrollo adaptarse a los cambios que puedan ocurrir en los requisitos. Trabajar con un alto grado de comunicación permite que el equipo (cliente-desarrollador) pueda tomar decisiones y aplicarlas inmediatamente. (*Fowler*, 2000)

A partir de las características analizadas y atendiendo a que se cuenta con un equipo de una persona, poco tiempo de desarrollo y un alto riesgo de realizar cambios en los requisitos del software, se selecciona la filosofía de las metodologías ágiles para guiar el proceso de desarrollo de la solución propuesta y dentro de las metodologías ágiles *Scrum* y la Programación Extrema (por sus siglas en inglés XP (*Xtreme Programming*)).

1.4.1 Scrum

Scrum es una metodología ágil que se puede usar para gestionar y controlar desarrollos complejos de software y productos usando prácticas iterativas e incrementales. Dicha metodología posee desventajas que no le permiten ser una opción factible para la propuesta de solución tales como: (*Kniberg*, 2007)

- Plantea un problema si el desarrollo está restringido por una fecha de entrega.

- Presupone que los requerimientos cambian, pero no de forma que el cliente acepte un diseño funcional/técnico.
- Se requiere de un experto en la metodología que monitorice su cumplimiento, es un inconveniente para personas con muy poca experiencia en el desarrollo de software.
- No posee ningún tipo de prácticas de ingeniería.

1.4.2 XP

La Programación Extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código.

Cuándo usar XP:(Ing.*Joskowicz*, 2008)

- Cuando los clientes no tienen idea clara de los requerimientos y los van cambiando.
- Para proyectos de riesgo: fecha fija de entrega, algo nunca hecho por el grupo, algo nunca hecho por la comunidad de desarrolladores.
- Cuando el equipo de desarrollo está integrado por 2 y 10 programadores. No es apto para proyectos con mucho personal.
- Integra gerentes y clientes a la formulación de preguntas, la negociación de cronograma y alcances, la creación de las pruebas.
- Automatiza las pruebas; es posible en casi todos los dominios. Es lícito repensar el diseño para facilitar el ensayo.

Características fundamentales de XP:(*Pressman*, 2010)

- **Desarrollo iterativo e incremental:** pequeñas mejoras, unas tras otras.
- **Pruebas unitarias continuas**: Se basan en las pruebas realizadas a los principales procesos con el objetivo de detectar futuros errores.(Gutierrez, y otros, 2010)
- **Integración del equipo de programación con el cliente o usuario**: Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- **Corrección de todos los errores:** Antes de añadir nueva funcionalidad. Hacer entregas frecuentes.

- **Refactorización del código:** Reescribir ciertas partes del código para aumentar su legibilidad y mantenimiento pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- **Uso de Metáforas:** La comunicación fluida es uno de los valores más importantes de XP, el hecho de incorporar al equipo una persona que represente los intereses del negocio y otras prácticas son valiosas entre otras cosas porque potencian enormemente la comunicación. Para conseguir que la comunicación sea fluida es imprescindible, entre otras cosas, utilizar el vocabulario del negocio.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Analizando los elementos anteriormente mencionados, así como el sistema informático que se quiere realizar y los recursos humanos con los que se cuenta. Se concluyó que es conveniente seleccionar la metodología XP dado a las facilidades que brinda esta para equipos de desarrollo con poco personal, donde se tiene una estrecha comunicación con el cliente y dominio de las tecnologías. Además la metodología seleccionada permite: Diseños sencillos del software y libre de complejidad, los cambios se implementan rápidamente tal y como fueron sugeridos, el manejo del cambio se convierte en parte sustantiva del proceso, elevando así su capacidad adaptativa al cambio, empleo de estándares de codificación y pruebas continuas.

1.5 Tendencias y tecnologías actuales. Selección de las herramientas y lenguajes de desarrollo.

Actualmente la información y el conocimiento son considerados importantes recursos para toda organización. El uso de las TIC juega el papel principal. La expansión de las mismas en todos los ámbitos de nuestra sociedad ha provocado transformaciones importantes en la vida económica y social de todas las personas. Los cambios en los medios de difusión de la información han sido notables.

El amplio desarrollo de las redes y medios de comunicación a nivel global, provocó un desarrollo acelerado de Internet, y con ello el desarrollo de aplicaciones para este espacio se vio beneficiado.

Muchas son las ventajas que posee desarrollar aplicaciones *web*, dentro de las que destacan: (Mora, 2002)

- El problema de gestionar el código en el cliente se reduce drásticamente, para introducir un cambio solo es necesario modificar la aplicación alojada en el servidor *web*.

- Se evitan problemas de inconsistencia en las actualizaciones, ya que no existen clientes con distintas versiones de la aplicación.
- Son independientes a las plataformas donde son ejecutadas, solo dependen de un navegador *web*.
- Facilitan la centralización de los datos.
- Las prestaciones requeridas en los ordenadores de los clientes no deben ser tan altas.
- En la mayoría de los casos son libres, sencillas de desarrollar y actualizar.
- Es por ello que las tecnologías utilizadas para el desarrollo de la propuesta de solución serán del tipo *web*.

1.5.1 Selección del framework de desarrollo en el servidor

Un *framework* de desarrollo es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. Su utilización fomenta la reutilización de código, promueve buenas prácticas de desarrollo y proporciona una reducción de tiempo en los procesos de desarrollo. (Alonso, 2006)

Zend Framework (ZF): Es un *framework* para desarrollo de aplicaciones *web* y servicios *web* con *PHP*, brinda soluciones para construir sitios *web* modernos, robustos y seguros. Además es de código abierto y trabaja con *PHP* 5. Está implementado usando código 100% orientado a objetos y la estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes.

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de ZF conforman un potente y extensible *framework* de aplicaciones *web* al combinarse. ZF ofrece una robusta implementación Modelo Vista-Controlador y una abstracción de base de datos fácil de usar. (Technologies, 2009)

Symfony: Está desarrollado completamente con *PHP*, proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. La última versión de este *framework* conocida es *Symfony* 2, entre sus principales características se encuentran. (*Symfony*.es, 2010)

- Desarrollo muy activo, puesto que incluye constantemente mejoras combinando la flexibilidad con la rapidez de ejecución.

- Amplia documentación.
- Reutilización muy activa de usuarios: la comunidad de usuarios crece cada día y ofrece un gran soporte de forma gratuita.
- Flexibilidad tanto en el diseño global del *framework*, como en su sistema de configuración y en los *plugins*.

Symfony 2 emplea por defecto **ORM Doctrine 2** (Mapeador de Objeto Relacional (*ORM*) por sus siglas en inglés). Una de las principales características de Doctrine es el bajo nivel de configuración que requiere para comenzar un proyecto. Puede generar clases a partir de una base de datos creada, y el programador puede especificar relaciones y agregar funcionalidades comunes para las clases generadas. No existe necesidad de generar o mantener esquemas complejos en lenguaje extensible de marcas (XML) por sus siglas en inglés. Otra característica es la habilidad de escribir consultas a la base de datos a partir de la programación orientada a objetos, llamada DQL (*Doctrine Query Language*). (Eguiluz, 2011)

La selección se basó en el conocimiento que se posee sobre el *framework Symfony*, evitando invertir tiempo y esfuerzo en el aprendizaje de una nueva tecnología de este tipo y aprovechar el mismo en el desarrollo de la propuesta de solución. Cabe destacar que en la UCI, *Symfony* es utilizado en varios proyectos reales, lo cual crea un cúmulo de conocimientos en la comunidad universitaria alrededor del mismo, que puede ser muy útil durante el proceso de desarrollo. La versión a utilizar será la 2.3.7, por ser la última actualización.

1.5.2 Selección del framework CSS Bootstrap

Bootstrap es un marco de trabajo que facilita la labor del diseño *web*, permitiendo crear interfaces *web* con *CSS* y *JavaScript*, permite diseñar la interfaz *web* dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, esto se denomina diseño adaptativo o diseño sensible. Ofrece todas las posibilidades de crear interfaces *web*, los diseños creados con *Bootstrap* son simples, limpios e intuitivos, esto les da agilidad a la hora de cargar y al adaptarse a otros dispositivos. El *framework Bootstrap* propone elementos con estilos predefinidos fáciles de configurar como: botones, menús desplegables, formularios, y otros. Contribuye con al ahorro de tiempo, debido a que es esta una de las tareas que más tiempo consume en el proceso de desarrollo de software. (LTD, 2014)

Se tuvo en cuenta para su selección: Su facilidad de uso, el ahorro considerable de tiempo y la existencia de conocimientos previos sobre esta tecnología. La versión a utilizar será la 2.3.2.

1.5.3 Lenguajes de programación

Java: El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc. con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC (*Personal Computer*) o *MAC's*[6]) y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma (*Waite*, 2001). El mismo elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. La filosofía de programación orientada a objetos es diferente a la programación convencional. (*Englander*, 2002)

- Java presenta características muy importantes que lo definen como alto candidato para dar solución a una parte de la propuesta de solución:
- Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable.
- Proporciona un conjunto de clases potentes y flexibles.
- Facilita la utilización de aplicaciones que se pueden incluir directamente en páginas *web* (*applets*).
- Presenta las siguientes ventajas: (Antonio, 2000)
- Robusto: El sistema de Java maneja la memoria de la computadora por ti. No te tienes que preocupar por apuntadores, memoria que no se esté utilizando, y otros. Java realiza todo esto sin necesidad de que uno se lo indique.
- Seguro: El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los *applets*, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.
- Portable: Como el código compilado de Java (conocido como *bytecode*) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computador que tenga implementado el intérprete de Java.

[6]MAC's: Ordenador personal diseñado, desarrollado, construido, comercializado y vendido por la compañía Apple Inc

- Independiente a la arquitectura: Al compilar un programa en Java, el código resultante un tipo de código binario conocido como *bytecode*. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.
- Multihilo: Un lenguaje que soporta múltiples hilos, es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo.
- Interpretado: Java corre en máquina virtual, por lo tanto es interpretado.
- Dinámico: Java no requiere que sean compiladas todas las clases de un programa para que este funcione. Al realizar una modificación a una clase, Java se encarga de realizar un *Dynamic Bynding* o un *Dynamic Loading* para encontrar las clases.

Se seleccionó el lenguaje *Java* para la implementación de estos algoritmos, por las características y ventajas analizadas en el epígrafe correspondiente a los lenguajes de programación, respaldado la selección la experiencia que tiene el equipo de desarrollo en el empleo de dicho lenguaje. La versión a utilizar del lenguaje Java será la 8.0 que es la más actualizada en los repositorios del sistema operativo *Ubuntu* 12.04 en la UCI.

PHP: Es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas *web* dinámicas, embebidas en páginas *HTML* y ejecutadas en el servidor. *PHP* no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión (*PHP*). (*Bakken*, y otros, 2001)

Al tratarse de software libre puede descargarse y utilizarse en cualquier aplicación personal o profesional de manera completamente libre. Además se dispone de acceso a su código fuente. *PHP* puede usarse en todos los sistemas operativos gratuitos del tipo *Unix*, como *Linux* o en las diferentes versiones de *Microsoft Windows*. (*PHP*.net, 2010)

La elección de *PHP* estuvo determinada por seleccionar *Symfony* como *framework* de desarrollo en el servidor, el cual se encuentra escrito en dicho lenguaje y la versión a utilizar será la 5.3.10, la misma es la más actualizada en los repositorios del sistema operativo *Ubuntu* 12.04 en la UCI. Sin embargo como el lenguaje *PHP* no es recomendable usarlo para la creación de algoritmos ávidos, ya que los mismos tienen un alto consumo de memoria y sobrecargan el servidor.

CSS: Las Hojas de Estilo en Cascada, *CSS* por sus siglas en inglés son complementos de código añadidos al *HTML* que se encargan de la apariencia del documento. Se utilizan para dar estilo a documentos *HTML* y *XML*, separando el contenido de la presentación. Este permite a los desarrolladores *Web* controlar el estilo y el formato de múltiples páginas *web* al mismo tiempo y que cualquier cambio en el estilo marcado para un elemento en la *CSS* afectará a todas las páginas vinculadas a la misma en las que aparezca el mismo. (Lancker, 2009) La versión 3 del lenguaje *CSS* ha incorporado valiosas novedades como son: (Luca, 2011)

- Nuevas alternativas para dibujar bordes con el uso de opciones tales como color, imágenes, y radio o redondeado.
- Novedades en el trabajo con fondos, con el uso de degradados y la posibilidad de incluir múltiples imágenes.
- Novedades en cuanto al uso del color y de la opacidad.
- Nuevas características para el trabajo con múltiples columnas.
- Novedades y algunos cambios en el uso de pseudo-elementos.
- Características relacionadas con la interfaz de usuario.
- Capacidad de rotación de elementos.
- Opciones de transformación de elementos.
- Incorporación de transición y también funciones de animación.

La selección de CSS3 como lenguaje para escribir el estilo visual de la propuesta de solución se hizo teniendo en cuenta que el mismo es un estándar determinado por la *World Wide Web Consortium* (*W3C*), y posee un amplio uso y difusión por parte de maquetadores, diseñadores y desarrolladores *web*.

HTML: Desde el surgimiento de internet se han publicado sitios *web* gracias al lenguaje *HTML*. Es un lenguaje estático para el desarrollo de sitios *web* (acrónimo en inglés de *Hyper Text Markup Language*, en español Lenguaje de Marcado de Hipertextos). Desarrollado por la *W3C*. Los archivos pueden tener las extensiones (htm, 2010).

Fue creado en 1986 por el físico nuclear Tim *Berners-Lee*; el cual tomó dos herramientas preexistentes: El concepto de Hipertexto (conocido también como *link* o ancla) el cual permite conectar dos elementos entre sí y el SGML (Lenguaje Estándar de Marcación General) el cual sirve para colocar etiquetas o marcas en un texto que indique cómo debe verse.

Es sencillo, permite describir hipertexto, el texto es presentado de forma estructurada y agradable. No necesita de grandes conocimientos cuando se cuenta con un editor de páginas *web*. Consta de archivos pequeños un rápido despliegue, es de fácil aprendizaje y lo admiten todos los exploradores. (Alvarez, 2009)

HTML constituye uno de los pilares sobre los que se asienta la *web*, es un lenguaje extensible al que se le pueden añadir características y funcionalidades mediante las *CCS* y *JavaScript* obteniéndose como resultado páginas *web* rápidas y sencillas.(W3C, 2014)

HTML 5 es la última versión conocida de este lenguaje, la misma está pensada con una mayor integración con los lenguajes CSS y *JavaScript*. *HTML* 5 ha revolucionado la *web* y no solo se ve como el presente, sino como el futuro, por las numerosas novedades que trae con respecto a la versión anterior, entre las que se encuentran la Inclusión de nuevas etiquetas que permiten representar elementos familiares de las páginas, tal es el caso de: <*header*> para el encabezado de las páginas, <*nav*> para la navegación, <*section*> para una sección de la página y <figure> para asignar un título a una imagen.

HTML 5 propone estándares para cada aspecto de la *web* y también un propósito claro para cada una de las tecnologías involucradas. *HTML* 5 provee los elementos estructurales, *CSS* 3 se encarga de volver esa estructura utilizable y atractiva a la vista y *JavaScript* tendrá el poder necesario para proveer dinamismo y construir aplicaciones *web* completamente funcionales y con mayor capacidad de interacción con el usuario. (*Gauchat*, 2012)

La selección de *HTML* 5 como el lenguaje para realizar el maquetado de la propuesta de solución está sustentada en las novedosas herramientas que incorpora para el desarrollo *web* y el hecho de ser este un estándar establecido por la *W3C*.

JavaScript: Este es un lenguaje interpretado, no requiere compilación. Utilizado principalmente en páginas *web*. Es similar a *Java*, aunque no es un lenguaje orientado a objetos, el mismo no dispone de herencias. La mayoría de los navegadores en sus últimas versiones interpretan código *JavaScript*. (Eguiluz, 2010)

El código JavaScript puede ser integrado dentro de nuestras páginas *web*. Para evitar incompatibilidades la W3C diseñó un estándar denominado DOM (en inglés *Document Object Model*, en su traducción al español Modelo de Objetos del Documento). (Eguiluz, 2010)

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos

como *Internet Explorer*, *Mozilla Firefox*, *Opera*, *Netscape*, entre otros, es el lenguaje de programación del lado del cliente más utilizado.

La selección de *JavaScript* como el lenguaje de script del lado del cliente, está determinada por el hecho de usar la librería *JQuery*, la cual está escrita en el lenguaje citado.

1.5.4 Entorno de desarrollo integrado

NetBeans IDE: Es un *IDE* de código abierto para desarrolladores. Posee todas las herramientas necesarias para crear aplicaciones *web*, de escritorio y por teléfonos móviles con los lenguajes de programación *Java*, C, C++ e incluso lenguajes dinámicos como: *PHP*, *JavaScript*, *Groovy*, y *Ruby*. Es fácil de usar, instalar y puede ser ejecutado en múltiples plataformas como *Windows*, *Linux*, *Mac* OS *X* y *Solaris*.

NetBeans 8.0 permite crear aplicaciones *web* con *PHP* 5, se puede escribir, compilar, depurar y ejecutar aplicaciones de escritorio y/o, *web* con múltiples lenguajes de programación, cuenta con un potente depurador integrado y además viene con soporte para *Symfony* y *AJAX*. (CORPORATION©, 2014)

Entre sus principales características se puede mencionar:

- Creación de proyectos PHP: Provee de una estructura para los proyectos que pueden crear junto a este IDE, además propone un esqueleto para organizar nuestro código fuente, el editor conjuntamente integra los lenguajes como *HTML*, *JavaScript* y *CSS*.
- Integración con *Symfony* y *Zend Framework*: Gracias a *NetBeans* ya es posible dejar de lado la consola de comandos de *Symfony* y centrarse en desarrollar en el IDE, además se encuentra cargadas todas las clases, ayuda en línea, y otros.
- Editor de código fuente: El editor de PHP, es mucho más ágil y a la vez robusto, contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la versión de PHP 5.3.
- Integración con *PHP Unit Testing*: Es posible crear test con *PHPUnit*, para diferentes funciones, luego realizar la comprobación y ver todos los resultados. En las propiedades *PHPUnit* puede definir una configuración personalizada de archivos *XML*, un archivo de arranque para las opciones de línea de comandos, o una serie de pruebas a medida, o puede que el *IDE* genera el código esqueleto para usted.

- Depuración de PHP: *NetBeans* integra muy bien la utilización Xdebug[7], gracias a esto es posible inspeccionar y examinar cada variable local, establecer puntos de interrupción y evaluar el código en nuestra lógica.
- Integración con *MySQL*: Posee una integración completa en términos de administración básico y avanzada de *MySQL*, y todo desde el mismo entorno.
- Integración con Sistemas de Control de Versiones: Esta es una de las condiciones necesarias para los proyectos y es la posibilidad de contar con la integración de sistemas de control de versiones, tales como SVN, CVS, Mercurial y Git.

El *IDE* de *NetBeans* para *PHP* también ofrece la línea de comandos de depuración: La salida del programa *PHP* aparece en una pantalla de línea de comandos en el *IDE* de sí mismo y se puede inspeccionar el código *HTML* generado sin tener que cambiar a un navegador.

La versión a utilizar es la 8.0, la cual fue lanzada el 18 de marzo de 2014 y es la más actualizada en los servidores de la UCI. La misma brinda un amplio soporte al desarrollo de aplicaciones *web* usando el *framework Symfony* y facilidades tanto para el completamiento de código, como para las tareas desarrolladas mediante líneas de comando. La selección fue respaldada por la familiarización y experiencia de trabajo con esta herramienta.

1.5.5 Herramientas CASE de modelado con UML

Las Herramientas CASE (por sus siglas en inglés *Computer Aided* Software *Engineering*, en español Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Estas herramientas respaldan todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.(León, 2000)

Visual Paradigm para UML: Es una herramienta CASE de modelado, que utiliza UML como lenguaje de modelado profesional y que soporta el ciclo de vida completo del desarrollo de software: Análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar ingeniería tanto directa como inversa.(visual-paradigm, 2013) El software de modelado UML aporta rapidez en la construcción

[7]Xdebug: Extensión que ofrece capacidades de depuración y perfilado.

de aplicaciones de calidad mejores y menores costos.(IBM, 2009) Soporta múltiples usuarios trabajando sobre el mismo proyecto y permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y la documentación del proyecto automáticamente en varios formatos como HTML, PDF y permite control de versiones por lo que se define como colaborativa. Destacan su robustez, usabilidad y portabilidad como principales características. Permite realizar ingeniería inversa a partir de bases de datos, soportando una amplia gama de bases de datos, entre las que se encuentran *Oracle*, *MySQL* y *PostgreSQL*. (visual-paradigm, 2013)

UML: El uso del lenguaje de modelado unificado (por sus siglas en inglés *UML*) se basa en especificar, visualizar y documentar los diferentes aspectos relativos a un sistema de *software* en desarrollo y para modelado de negocios y almacenamiento de datos. Está destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que fue diseñado para modelar cualquier tipo de proyectos, tanto informáticos, de arquitectura, o de cualquier otra rama. UML posibilita el establecimiento de una serie de requerimientos y estructuras necesarias para plasmar un sistema de *software* previo al proceso intensivo de escribir código y sus beneficios son directamente proporcionales a la complejidad del sistema que se desea. Intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos se crea una documentación que cualquier desarrollador con conocimientos de UML será capaz de entender. (Ivar Jacobson, 2000)

Como lenguaje de modelado se decide usar el UML empleando la herramienta *Visual Paradigm* para *UML* porque realiza de una forma íntegra los planes de construcción del *software* a desarrollar, soporta el ciclo de vida completo del desarrollo del *software* a obtener, posee la capacidad de ejecutarse sobre diferentes sistemas operativos y permite realizar ingeniería inversa a partir de bases de datos. La versión a utilizar será la 8.0, la misma es la más actualizada en los servidores de la UCI.

1.5.6 Sistema gestor de base de datos

Un Sistema Gestor de Bases de Datos (SGBD o DBMS (del inglés, *Data Base Management System*) es un software específico cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Es un software que permite introducir, organizar y recuperar la información de las bases de datos. Existen distintos tipos de gestores de bases de datos: Relacional, jerárquico, y otros. El modelo relacional es el utilizado por casi todos los gestores de bases de datos para PC´s.

Algunos ejemplos de SGBD son *PostgreSQL*, *MySQL*, entre otros.

PostgreSQL: Es un servidor de base de datos relacional orientada a objetos de software libre, soporta casi toda la sintaxis *SQL* y ofrece muchas características modernas tales como: Consultas complejas, integridad referencial, integridad transaccional, control de concurrencia multi-versión, entre otras. También soporta almacenamiento de objetos grandes (imágenes, sonido y video). (PostgreSQL, 2012)

Una de las características que comparte con otros motores de bases de datos es el hecho de ser multiplataforma y tener varias herramientas para administrar. Cuenta con herramientas gráficas como *PgAdmin* y *phpPgAdmin*, las cuales hacen que la administración de la base de datos sea sencilla. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo al mismo tiempo al sistema, sin embargo presenta lentitud en caso contrario. Debido a su licencia libre *PostgreSQL* puede ser utilizado, modificado y distribuido por todo el mundo de forma gratuita para cualquier fin, ya sea privado, comercial o académico. (Gerrero, 2013)

MySQL: Es un sistema para la administración de bases de datos relacional (RDBMS) rápido y sólido creado por la empresa sueca *MySQL* AB. Es un sistema gestor de base de datos relacional donde los datos están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación *SQL*. El código fuente de *MySQL* se puede descargar y está accesible a cualquiera, por otra parte, usa la licencia GPL (GNU *General Public License*) para aplicaciones no comerciales.

MySQL es muy utilizado en aplicaciones *web*. Es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional *MyISAM*, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones *web* hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a *MySQL* ideal para este tipo de aplicaciones. (Mikoluk, 2013)

La facilidad del manejo de las bases de datos *MySQL*, la correcta integración que brinda en ambientes de desarrollo en sistemas operativos *GNU/Linux*, su soporte para el lenguaje de programación *PHP*, su carácter libre y el conocimiento previo sobre la misma hicieron que fuera seleccionado como el gestor de bases de datos para el desarrollo de la propuesta de solución. La versión a utilizar será la 5.5 la misma es la más actualizada en los repositorios del sistema operativo Ubuntu 12.04 en la UCI.

1.5.7 Servidor web Apache

Servidores w*eb*:

- En Internet, un servidor es un ordenador remoto que provee los datos solicitados por parte de los navegadores de otras computadoras.
- En redes locales se entiende como el software que configura un PC como servidor para facilitar el acceso a la red y sus recursos.

Los Servidores almacenan información en forma de páginas *web* y a través del protocolo *HTTP* lo entregan a petición de los clientes (navegadores *web*) en formato *HTML*. Uno de los servidores *web* más difundidos a nivel mundial es Apache.

El servidor *web* apache no solo funciona en la mayoría de las versiones de *Unix* sino que, además, funciona en *Windows 2000/NT/9x* y en muchos otros sistemas operativos de escritorio y de tipo servidor como son Amiga OS 3.x y OSl2. Es una tecnología gratuita de código fuente abierta. (*Wheeler*, 2008)

No posee una interfaz de usuario gráfica para su administración, mediante un sencillo archivo de configuración llamado httpd.conf se puede configurar. Soporta tanto host basados en IP como host virtuales. Brinda una gran flexibilidad en el registro y la monitorización del estado del servidor mediante un navegador *web*.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código. El resto de funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar para añadir un conjunto de utilidades al servidor sin volver a instalar el software. (Ciberaula.com, 2008)

La facilidad de configuración, la correcta integración que brinda este servidor en un ambiente de desarrollo en sistemas operativos *GNU/Linux* y su amplia difusión y uso en la red de redes, lo hicieron convertirse en el servidor *web* a utilizar en el desarrollo de la propuesta de solución. La versión a utilizar será la 2.2, la misma es la más actualizada en los repositorios del sistema operativo *Ubuntu* 12.04 en la UCI.

1.6 Conclusiones del capítulo

El estudio realizado sobre el estado del arte relacionado con el proceso de informatización del SGOE, permitió definir los conceptos necesarios para complementar la investigación. El análisis de soluciones homólogas contribuyó a una mayor comprensión de los aspectos comunes entre estas, quedando evidenciada la necesidad de desarrollar un nuevo sistema, nutriéndose de algunas de las

características de las aplicaciones estudiadas. Además permitió seleccionar las herramientas y tecnologías a utilizar en el desarrollo de la propuesta de solución. Quedando seleccionada la metodología XP para guiar el proceso de desarrollo, como lenguaje de modelado fue elegido *UML* y como herramienta *CASE* para realizar el modelado del software se optó por *Visual Paradigm* para *UML* en su versión 8.0. Tras determinarse que la propuesta de solución será una aplicación *web*, las tecnologías para su desarrollo se distinguen por su uso en el servidor y en el cliente.

En el lado del servidor, los lenguajes a utilizar serán PHP en su versión 5.3.10, este lenguaje será utilizado aprovechando las potencialidades que provee el *framework* de desarrollo *Symfony* en su versión 2.3.7 y Java 8.0 para la implementación de algoritmos evitando de esta forma la sobrecarga del servidor, alto consumo de memoria y aumento del tiempo de respuesta.

En el lado del cliente será utilizado HTML en su versión 5 como lenguaje de maquetado, como lenguaje para proveer estilos visuales se empleará CSS en su versión 3, este lenguaje será usado a través del *framework* de diseño *Bootstrap* en su versión 2.2.2, como lenguaje de scripting en el cliente será empleado JavaScript a través de la librería *JQuery* en su versión 1.8.3, aprovechando el uso de esta librería se empleará en la construcción de las interfaces de usuario. En el acceso a la base de datos será utilizado el ORM Doctrine en su versión 2.0. Para el almacenamiento de la información se determinó a *MySQL* en su versión 5.3 como gestor de bases de datos. El servidor *web* a utilizar será Apache en su versión 2.2.22 y por sus disímiles características que lo hacen un potente IDE fue seleccionado *NetBeans* en su versión 8.0 para desarrollar la propuesta de solución.

CAPÍTULO 2: EXPLORACIÓN Y PLANIFICACIÓN

En el presente capítulo se documentarán las fases de exploración y planificación. Se describirá la propuesta de solución que dará respuesta al problema planteado. Se definirán los principales conceptos asociados al dominio del problema, representando en un diagrama conceptual del negocio la relación entre ellos. Se especifican los roles que interactuarán con el sistema y serán generados los artefactos que propone para esta etapa la metodología XP, tales como: historias de usuario (HU)[8], estimación de esfuerzos por historias de usuario, plan de entregas e iteraciones.

2.1 Descripción de la propuesta de solución

La propuesta de solución tiene como objetivo hacer más eficiente el proceso de planificación y control del SGOE en la Facultad 4 de la UCI. Dentro del sistema los usuarios podrán desempeñar los siguientes roles: Súper-Administrador, Administrador y Estándar. Inicialmente el acceso se realizará mediante el usuario con rol Súper-Administrador que contiene la aplicación en su configuración predeterminada, este usuario es único en el sistema y es el encargado de ingresar en el sistema los datos de los usuarios.

El usuario con rol Súper-Administrador podrá:

- Importar a la base de datos del sistema los datos del personal a tener en cuenta en la planificación de la guardia.
- Definir los roles de cada usuario ingresado en el sistema.

El usuario con rol Administrador será el encargado de:

- Gestionar la planificación del SGOE.
- Replanificar el SGOE.
- Controlar la asistencia.
- Crear las brigadas[9] de estudiantes.
- Cambiar el estado de disponibilidad de cada usuario del sistema.

[8]HU: Describen detalladamente los requisitos funcionales del *software*

[9]Una brigada está integrada por 3 o más personas, una brigada realiza su guardia un día especifico.

- Exportar a formato PDF [10]la planificación general del SGOE y la asistencia diaria.
- Importar a la base de datos del sistema los datos del personal a tener en cuenta en la planificación de la guardia.
- Mostrar planificación del SGOE.
- Crear reportes[11] de cumplimiento del SGOE.

El usuario con rol Estándar podrá:

- Mostrar su planificación del SGOE.
- Mostrar su asistencia.
- Realizar intercambios de turnos en el SGOE.
- Autorizar intercambio de turnos en el SGOE.
- Mostrar información referente al SGOE.
- Conocer a que brigada pertenece una persona determinada.

Las acciones que realizan los usuarios con rol Estándar pueden ser realizadas por los usuarios con rol Administrador, ya que el rol Administrador fue definido en la descripción de la propuesta de solución como una especialización del rol Estándar.

El sistema podrá ser desplegado de manera local en la computadora de la persona encargada en la Facultad de realizar la planificación y control del SGOE, en la cual confluirán al mismo tiempo el servidor de base de datos y de aplicaciones.

2.2 Requisitos no funcionales del sistema

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener y que de una u otra forma puedan limitar el sistema. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (bligoo, 2014)

A continuación se hace alusión a los requisitos no funcionales:

Requerimientos de software:

[10] PDF: Portable Document Format (formato de documento portátil).

[11] Reporte: Es una visualización de la asistencia de los usuarios en una fecha determinada.

Debe poder visualizarse en los navegadores:

- Mozilla Firefox 5.0 o superior.
- Internet Explorer 9 o superior.
- Google Chrome 10.0 o superior.
- Opera 11.0 o superior.
- Safari 5.0 o superior.

Requerimientos de hardware:

Para las computadoras del cliente:

- Se requiere tengan tarjeta de red.
- Se requiere tengan al menos 256 MB de memoria RAM.
- Procesador Pentium 500 MHz como mínimo.

Para el servidor:

- Se requiere tarjeta de red.
- Se requiere tenga al menos 1GB de RAM.
- Procesador Pentium 1.3 GHz como mínimo.

Requerimientos de Seguridad:

- Se podrá acceder al sistema solamente después de autenticarse (usuario y contraseña).
- Los usuarios que pueden acceder son los que estén en el dominio UCI y registrados en la base de datos.

Requerimiento de Usabilidad:

- La interfaz de usuario de la aplicación debe ser sencilla, intuitiva, que le permita al usuario que interactúa con el sistema realizar cualquier labor en el menor tiempo posible y con pocas acciones.

Requerimientos de Rendimiento:

- Los tiempos de respuestas del sistema en el procesamiento de datos no deben exceder de los 10 segundos.

Requerimientos Políticos Culturales:

- El sistema no debe contener palabras en otros idiomas.
- El sistema debe respetar los términos empleados normalmente por los especialistas en el tema de la esfera que se automatiza.

Requerimientos Legales:

- Las herramientas de desarrollo son libres.
- Reconocido y autorizado por instancias superiores tales como la dirección de producción de la UCI (Facultad 4).
- Documentación legal de uso como Declaración de Autoría.

Requerimientos de Confiabilidad:

- Los datos que se obtendrán de la aplicación deben ser 100% precisos.
- Deben establecerse los mecanismos necesarios para el restablecimiento del sistema ante fallos de comunicación u otros.

Ya definidos los requisitos no funcionales que debe cumplir el sistema y haciendo uso de ellos se comienza la etapa del modelado del sistema.

2.3 Usuarios relacionados con el sistema

Los usuarios del sistema son representados mediante entidades que hacen uso a través de una conexión de red de los recursos proporcionados por un sistema, atendiendo a los diferentes perfiles de usuarios y siempre con las restricciones lógicas definidas para cada una de estas entidades al interactuar con dicho sistema, permitiendo definir niveles de accesibilidad y una mayor conservación de las propiedades que posea el sistema. (Escribano, 2002)

Tabla 1 Usuarios del sistema

Usuario	Descripción
Súper-Administrador	Ingresa en el sistema los datos de los usuarios y define el rol de cada usuario el sistema

Administrador	Modifica y realiza todas las funcionalidades definidas para el sistema, pudiendo controlar y chequear la información de todos los usuarios del sistema.
Estándar	Consulta los datos referentes a la planificación de su guardia, ver su registro de asistencia, intercambiar su turno de guardia y consultar brigada de la guardia a la que pertenece.

2.4 Diagrama conceptual del negocio

El modelo de dominio se describe mediante diagramas UML. Un diagrama conceptual del negocio no es más que un artefacto construido bajo las reglas de UML durante la concepción de un proyecto informático. Este modelo puede ser utilizado para capturar y expresar el entendimiento ganado sobre el negocio. (Fowler, 2000)

Se generó el artefacto que se muestra en la Ilustración 1 para un mejor entendimiento de la propuesta de solución a desarrollar, a partir de identificar los conceptos y objetos relacionados con la planificación y control del SGOE en la Facultad 4 de la UCI.

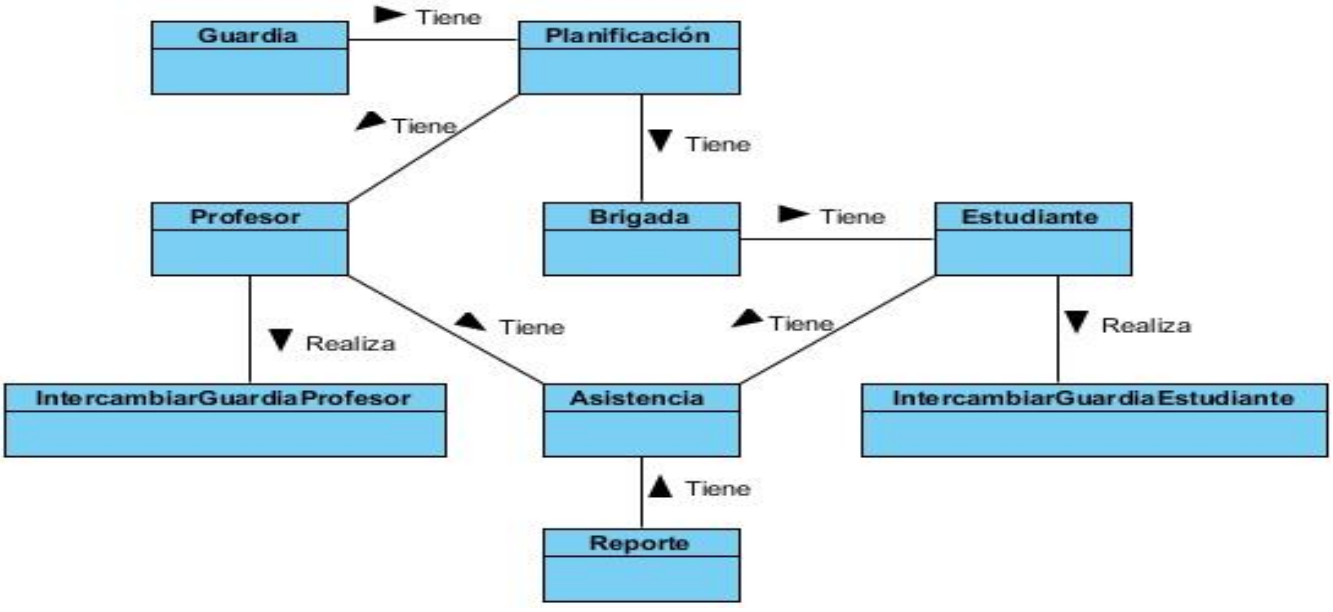


Ilustración 1 Diagrama conceptual

2.5 Exploración

En la fase de exploración de la metodología XP se definen las historias de usuario, que son la descripción de las funcionalidades con que debe cumplir el sistema. Describiendo en una plantilla las características que deben ser adicionadas al programa. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas y las tecnologías que utilizarán en el desarrollo del proyecto (Ing.Joskowicz, 2008).

2.5.1 Historias de usuario

Las HU describen las necesidades que el software debe solventar. El cliente describe brevemente las características que el sistema debe poseer, empleando terminologías no técnicas. Se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, presiden la creación de las pruebas de aceptación. Deben ser elaboradas lo suficientemente comprensibles y son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración. (Escribano, 2002)

A continuación se muestran las funcionalidades del sistema y las HU para cada una de ellas:

1. Autenticar usuario.
2. Ingresar datos del usuario.
3. Crear brigadas del SGOE.
4. Generar planificación del SGOE.
5. Replanificar SGOE.
6. Mostrar planificación del SGOE.
7. Mostrar mi planificación del SGOE.
8. Crear reporte de cumplimiento del SGOE.
9. Mostrar reporte de cumplimiento del SGOE.
10. Mostrar asistencia al SGOE.
11. Buscar integrantes en el SGOE para solicitar intercambio de turnos.
12. Intercambiar turnos en el SGOE.
13. Autorizar intercambio de turnos en el SGOE.
14. Listar intercambio de turnos en el SGOE.
15. Mostrar leyenda.

Tabla 2 Autenticar usuario

Historia de usuario	
Número: 1	**Usuario:** Estándar
Nombre: Autenticar usuario	
Prioridad en el negocio: Alta	**Riesgo de desarrollo:** Medio
Puntos de estimación:1	**Iteración:**1
Programador responsable: José Angel Díaz Romero	
Descripción: El sistema comprueba que los datos introducidos por los usuarios para acceder a este son correctos.	

Tabla 3 Actualizar datos del usuario

Historia de usuario	
Número:2	**Usuario:** Súper-Administrador
Nombre: Actualizar datos del usuario	
Prioridad en el negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos de estimación: 1	**Iteración:**1
Programador responsable: José Angel Díaz Romero	
Descripción: El sistema debe insertar todos los usuarios pertenecientes a la facultad.	

Tabla 4 HU Crear brigada del SGOE

Historia de usuario	
Número: 3	Usuario: Administrador
Nombre: Crear brigadas del SGOE	

Prioridad en el negocio: Alta	Riesgo de desarrollo: Alta
Puntos de estimación: 2	Iteración: 1
Programador responsable: José Angel Díaz Romero	
Descripción: El sistema brindará la posibilidad al usuario Administrador de crear las brigadas, las mismas estarán compuestas por no menos de 3 integrantes.	

Tabla 5 Generar planificación del SGOE

Historia de usuario	
Número: 4	Usuario: Administrador
Nombre: Generar planificación del SGOE	
Prioridad en el negocio: Alta	Riesgo de desarrollo: Alto
Puntos de estimación: 2	Iteración: 1
Programador responsable: José Angel Díaz Romero	
Descripción: El sistema tendrá la posibilidad de generar automáticamente la planificación de la guardia obrero estudiantil en la Facultad 4. Algunos de los parámetros a tener en cuenta son: la categoría docente, el sexo, si reside en la universidad (si no es reside asignar un local para su estadía en el centro durante el cumplimiento del SGOE) y el estado de disponibilidad (si el usuario está enfermo o presenta alguna anomalía, salidas urgentes por cuestiones justificadas y vacaciones que le impida la correcta prestación del SG, además en el caso de que el usuario sea de sexo femenino tener en cuenta si está embarazada o tiene niños pequeños que le impidan estar disponible para prestar el servicio). Las mujeres no pueden hacer el último turno de guardia en ninguna de las zonas.	

2.6 Planificación

XP plantea la planificación como un constante diálogo entre el cliente y los desarrolladores donde los primeros definen el alcance del proyecto, la prioridad de las HU, la composición de las versiones y la fecha de entrega de las mismas, y los desarrolladores estiman el tiempo necesario para implementar cada HU, el cual debe ser de entre una y tres semanas. En caso de extenderse más de tres semanas la misma debe ser modificada o dividida en otras más pequeñas. Una duración de menos de una semana indica que es demasiado sencilla y deberá ser agrupada. (Fowler, 2000)

Esta fase genera artefactos de suma importancia para el desarrollo del proyecto, como el Plan de Iteraciones y el Plan de Entregas, los cuales se muestran a continuación.

2.6.1 Estimación de esfuerzo por historias de usuario

A continuación se muestra la estimación del esfuerzo por cada HU involucrada en el desarrollo de la aplicación:

Tabla 6 Estimación de esfuerzo por HU

Historia de usuario	Puntos de estimación
Autenticar usuario	1
Actualizar datos del usuario	1
Crear brigadas del SGOE	2
Generar planificación del SGOE	2
Replanificar SGOE	1
Mostrar planificación del SGOE	1
Mostrar mi planificación del SGOE	1
Crear reporte de cumplimiento del SGOE	1
Mostrar reporte de cumplimiento del SGOE	1
Mostrar asistencia al SGOE	1
Buscar integrantes en el SGOE para solicitar intercambio de Turnos	1
Intercambiar turnos en el SGOE	1
Autorizar intercambio de turnos en el SGOE	1
Listar intercambio de turnos en el SGOE	1
Mostrar leyenda	1

2.6.2 Plan de entregas

El plan de entregas deber ser lo más real posible porque constituye un documento oficial por el cual los clientes le exigen a los desarrolladores la entrega de las distintas versiones del producto. (Escribano, 2002) El equipo de desarrollo de la propuesta de solución a partir de una reunión con el cliente, teniendo en cuenta las prioridades planteadas por el mismo y las estimaciones realizadas por los desarrolladores, definió el siguiente Plan de Entregas:

Tabla 7 Plan de entregas

Entregable	1ra entrega (1ra semana de Abril)	2da entrega (2da semana de Mayo)	3ra entrega (3ra semana de Junio)
Sistema de planificación y control del servicio de guardia obrero-estudiantil.	Versión 0.1	Versión 0.2	Versión 0.3

Tabla 8 Funcionalidades disponibles por entrega del producto

Historia de Usuario	1ra entrega	2da entrega	3ra entrega
HU1. Autenticar usuario	X	X	X
HU2. Actualizar datos del usuario	X	X	X
HU3. Crear brigadas del SGOE	X	X	X
HU4. Generar planificación del SGOE	X	X	X
HU5. Replanificar SGOE	X	X	X
HU6. Mostrar planificación del SGOE		X	X
HU7. Mostrar mi planificación del SGOE		X	X
HU8. Crear reportes de cumplimientos del SGOE		X	X
HU9. Mostrar reportes de cumplimiento del SGOE		X	X
HU10. Mostrar asistencia al SGOE		X	X
HU11. Buscar integrantes en el SGOE para solicitar intercambio.			X
HU12. Intercambiar turnos en el SGOE			X
HU13. Autorizar intercambio de turnos del SG			X
HU14. Listar intercambios de turnos en el SGOE			X

HU15. Mostrar leyenda			X

2.6.3 Iteraciones

La metodología XP en su filosofía propone iteraciones de aproximadamente tres semanas de duración y para cada iteración se define un conjunto de HU que se van a implementar. En la primera iteración se seleccionan las HU que abarquen los aspectos más importantes de la arquitectura global, es decir, las HU que hagan referencia a la construcción de la estructura de todo el sistema. (Ing.Joskowicz, 2008)

En XP una iteración no necesariamente termina con una entrega del producto, por lo que una versión puede implicar varias iteraciones del proceso de desarrollo. Permitiendo al equipo de proyecto ubicar las historias de usuarios previstas para dicha versión, en tantas iteraciones como estime conveniente. (Fowler, 2000)

La elaboración del plan de entrega y la estimación de esfuerzo por cada HU permitieron diseñar un plan de iteraciones que regirá el orden en el que se implementarán cada una de las HU. La Tabla 18 deja constancia del mismo.

Tabla 9 Planificación de las iteraciones

Iteraciones	Orden de las HU a implementar	Cantidad de tiempo de trabajo
Iteración 1	HU1. Autenticar usuario HU2. Actualizar datos del usuario HU3. Crear brigadas del SGOE HU4. Generar planificación del SGOE HU5. Replanificar SGOE	7 semanas
Iteración 2	HU6. Mostrar planificación del SGOE HU7. Mostrar mi planificación del SGOE HU8. Crear reportes de cumplimientos del SGOE HU9. Mostrar reportes de cumplimiento del SGOE HU10. Mostrar asistencia al SGOE	5 semanas

Iteración 3	HU11. Buscar integrantes en el SGOE para solicitar intercambio. HU12. Intercambiar turnos en el SGOE HU13. Autorizar intercambio de turnos del SGOE HU14. Listar intercambios de turnos en el SGOE HU15. Mostrar leyenda	5 semanas

En la fase de planeación de la entrega de la metodología programación extrema se definen varias iteraciones sobre el sistema antes de ser entregado. Los elementos que deben tomarse en cuenta durante la elaboración del plan de la iteración son: Historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación. (Joskowicz, 2008)

2.7 Conclusiones del capítulo

Durante el desarrollo del capítulo que concluye se expusieron los artefactos generados por las dos primeras fases de la metodología XP, asegurando una implementación por etapas correctamente descritas y detalladas. En estas fases emergen factores fundamentales para el desarrollo del software, como el diálogo entre el cliente y los desarrolladores. Además se fundamentaron las HU que servirán como base para la construcción del sistema, se priorizaron teniendo en cuenta la importancia en la lógica de negocio que engloban los requisitos funcionales y fueron definidas en que iteraciones deberán ser implementadas. Se generó el Plan de entregas con las fechas en las que será liberada cada versión del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

La metodología XP plantea que la implementación debe realizarse de forma iterativa e incremental, lo que permite que las funcionalidades desarrolladas en esta fase generen al final de cada una un entregable funcional que implementa la HU asignadas a la iteración. Se define la arquitectura de la propuesta de solución. Se analizan los estándares de codificación y los patrones de diseño a usarse durante el proceso de implementación del software. Son determinadas las tarjetas clase-responsabilidad-colaboración (CRC), representando las clases que se implementarán en el sistema. Cada HU es dividida en tareas de ingeniería y se somete la propuesta de solución a un proceso riguroso de pruebas.

3.1 Descripción de la arquitectura

La arquitectura de la propuesta de solución está determinada por el uso del *framework Symfony* 2 en su desarrollo, el cual permite el desarrollo de aplicaciones *web* siguiendo una arquitectura Modelo-Vista-Controlador (MVC). El patrón arquitectónico MVC separa el código en tres capas: (Eguiluz, 2011)

La capa del Modelo define la lógica de negocio (la base de datos pertenece a esta capa). Es aquí donde actúan los ORM *Propel* y *Doctrine*, permitiendo al desarrollador abstraerse de la base de datos. Las clases relacionadas con el modelo se guardan en los directorio SPCGOE/UsuarioBundle/Entity/ y SPCGOE/GuardiaBundle/Entity/ del proyecto.

La Vista es con lo que el usuario interactúa (un motor de plantillas es parte de esta capa), o sea, es principalmente la capa de plantillas PHP. Estas son guardadas en los directorios SPCGOE/GuardiaBundle/Resources/views y SPCGOE/GuardiaBundle/Resources/viewsdentro de las aplicaciones del proyecto.

El Controlador es la pieza de código que llama al Modelo para obtener algunos datos que le pasa a la Vista para la presentación al cliente. Todas las solicitudes son gestionadas por los controladores. Las clases relacionada con los controladores se guardan en los directorios SPCGOE/UsuarioBundle/Controller y SPCGOE/GuardiaBundle/Controller del proyecto.

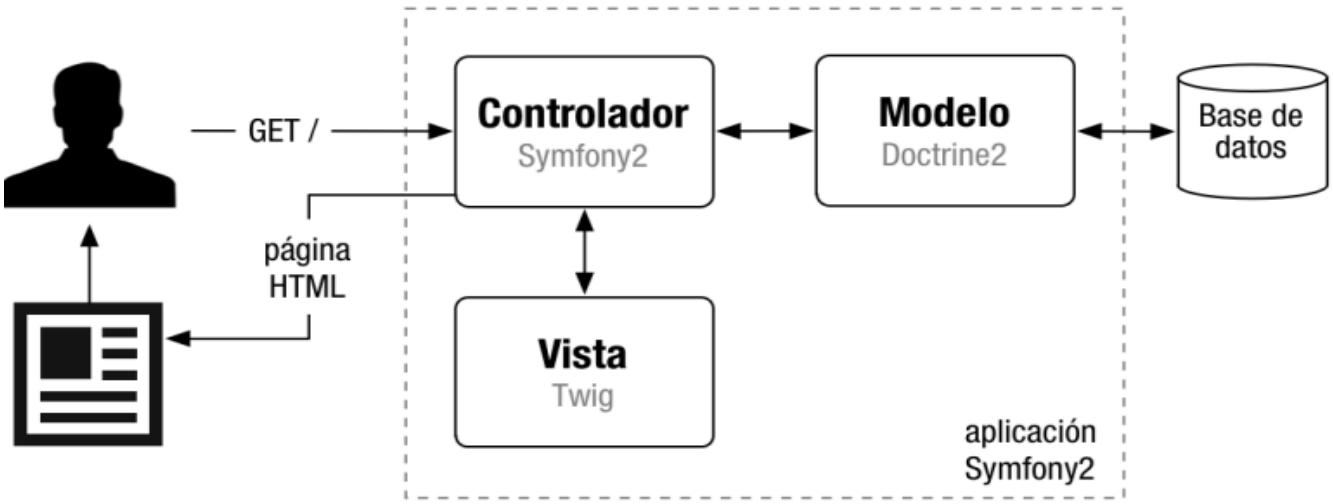


Ilustración 2 Patrón MVC

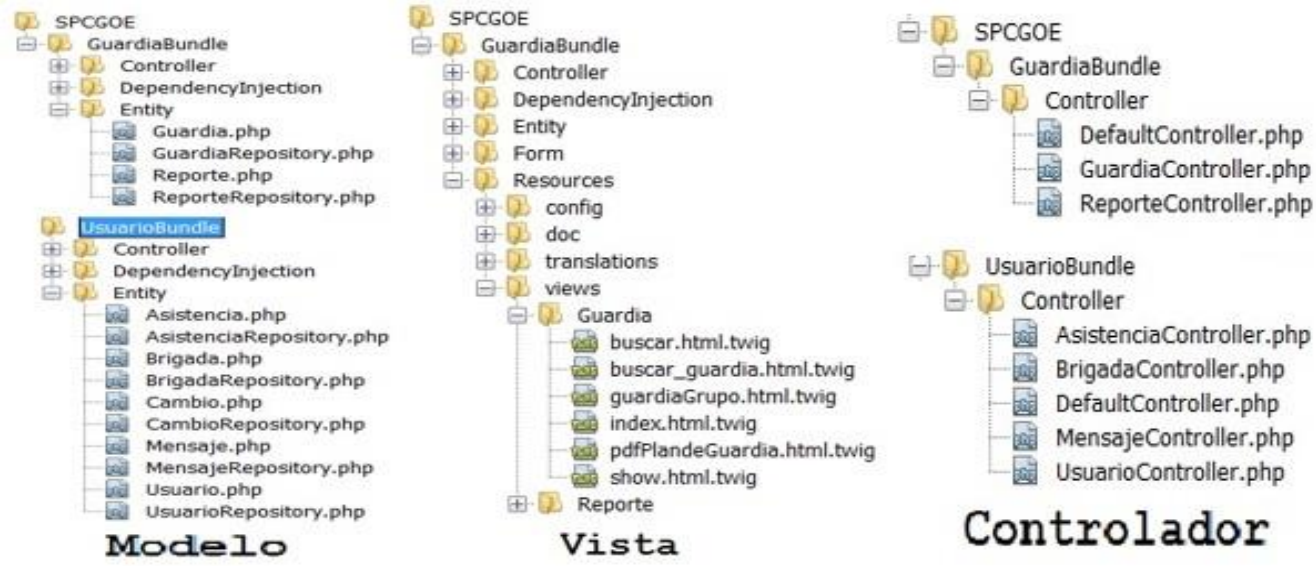


Ilustración 3 Ejemplo del patrón MVC

3.2 Tarjetas CRC

Las tarjetas CRC son un modelo simple, fácil de comprender y aplicar. En las mismas quedan plasmadas las responsabilidades de la clase, que no son más que su propósito y también representan las dependencias o colaboraciones de otras clases para realizar sus responsabilidades. (Escribano, 2002)

A continuación se muestra las tarjetas CRC correspondiente a la propuesta de solución:

Tabla 10 Tarjeta CRC Guardia

Tarjeta CRC	
Clase: Guardia	
Responsabilidades: Es la clase encargada de gestionar todo lo referente al SG (Planificación, modificación y control).	**Colaboraciones:** Brigada, Usuario.

Tabla 11 Tarjeta CRC Brigada

Tarjeta CRC	
Clase: Brigada	
Responsabilidades: Clase que registra todos los integrantes de cada brigada.	**Colaboraciones:** Usuario, Guardia.

Tabla 12 Tarjeta CRC Usuario

Tarjeta CRC	
Clase: Usuario	
Responsabilidades: Es la clase encargada de gestionar todo lo referente a los usuarios (estudiantes y profesores).	**Colaboraciones:** Brigada, Asistencia, Mensaje, Cambio, Reporte, Guardia.

Tabla 13 Tarjeta CRC Reporte

Tarjeta CRC	
Clase: Reporte	
Responsabilidades: Es la clase encargada de guardar la situación existente y registrada del SG en cuanto a cumplimiento, además de generar documentos (reportes de cumplimiento) en formato PDF.	**Colaboraciones:** Asistencia.

Tabla 14 Tarjeta CRC Asistencia

Tarjeta CRC	
Clase: Asistencia	
Responsabilidades: Es la clase encargada de guardar la asistencia de todos los usuarios del sistema al SGOE.	**Colaboraciones:** Usuario, Guardia, Reporte.

Tabla 15 Tarjeta CRC Mensajes

Tarjeta CRC	
Clase: Mensaje	
Responsabilidades: Es la clase encargada de registrar los mensajes de solicitud de intercambio de turnos del SGOE enviados o recibidos por un usuario.	**Colaboraciones:** Usuario, Cambio.

Tabla 16 Tarjeta CRC Cambio

Tarjeta CRC	
Clase: Cambio	
Responsabilidades: Es la clase encargada de registrar los intercambios de turnos del SGOE realizados por dos usuarios.	**Colaboraciones:** Usuario.

3.3 Diagrama entidad relación

El Diagrama Entidad Relación (DER) mostrado en la ilustración 4 representa el modelo físico correspondiente a la base de datos de la propuesta de solución, el mismo fue generado de forma automática por la herramienta *CASE Visual Paradigm* para *UML*. (Mikoluk, 2013)

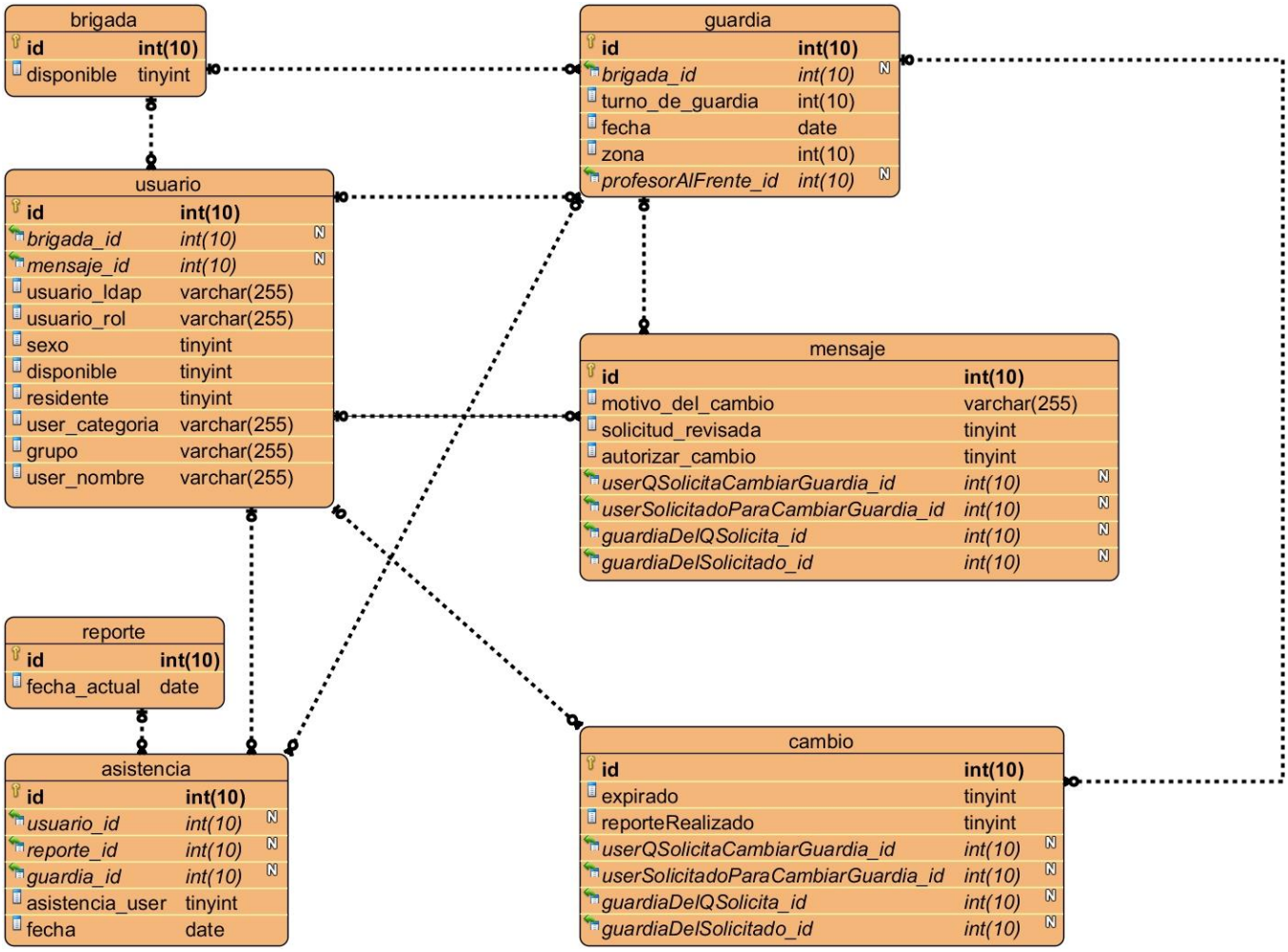


Ilustración 4 Diagrama entidad relación

3.4 Patrones de diseño

La ventaja fundamental de la utilización de *frameworks* para el desarrollo de aplicaciones es que los mismos incluyen patrones de diseño en su arquitectura de forma nativa, entre ellos los patrones: Patrones Generales de Software para Asignar Responsabilidades (del inglés *General Responsibility Assignment* Software *Patterns* (GRASP)) y Grupo de 4 (del inglés *Gang of Four* (GoF)). (Gamma, 1995)

Patrones de diseño GRASP

Es un conjunto de patrones que representa los principios básicos de la asignación de responsabilidades. A continuación se muestran los patrones GRASP empleados para el diseño del sistema. (Eguiluz, 2011)

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. El mismo se evidencia en la acción reporteDiarioAction() de la clase controladora ReporteController.php cuando se crea un objeto de Tipo Asistencia para asignársela a un usuario determinado.

```
public function reporteDiarioAction() {
    $req = $this->getRequest();

    $em = $this->getDoctrine()->getManager();
    $user = $this->getUser();
    $minombre = $user->getUsername();
    $usuario = $em->getRepository('SPCGOEUsuarioBundle:Usuario')->Buscaruserbrigada($minombre);
    if (count($usuario) == 0) {
        $rol_user = "";
    } else {
        $rol_user = $usuario[0]->getUsuarioRol();
    }
    $fecha_request = $req->get('fecha_reporte');
    $fecha = new \DateTime($fecha_request);
    $esta_reporte = $em->getRepository('SPCGOEGuardiaBundle:Reporte')->BuscarReporte($fecha);
    if (!$esta_reporte) {
        //$guardia = $em->getRepository('SPCGOEGuardiaBundle:Guardia')->finduserDiaAnteriror($fecha);
        $guardia = $em->getRepository('SPCGOEUsuarioBundle:listaGuardia')->findAllGuardiaByFecha($fecha);
        foreach ($guardia as $lista) {
            $user = $lista->getUsuario();
            $user_guardia = $lista->getGuardia();
            $asistencia = new Asistencia();
            $asistencia->setAsistenciaUser(true);
            $asistencia->setGuardia($user_guardia);
            $asistencia->setUsuario($user);
            $asistencia->setFecha($fecha);
            $em->persist($asistencia);

        }
```

Ilustración 5 Ejemplo de utilización del patrón creador

Bajo Acoplamiento: Este patrón plantea que la dependencia entre las clases que conforman la arquitectura del sistema debe ser mínima, debido a que a la hora de realizar modificaciones en la clase, no afecte la estructura de las restantes. En la aplicación se concibió mantener separadas las clases pertenecientes al modelo de datos, las vistas de usuario y las clases controladoras.

Alta Cohesión: Este patrón plantea que las clases se deben apoyar del funcionamiento de otras, para lograr su objetivo; y no incluir en ella misma todo el proceso del negocio, ya que sería difícil de comprender, reutilizar y le afectarían constantemente los cambios. Esto se evidencia en la aplicación en el método createGuadiaList() para crear la lista de guardia de cada usuario, utiliza la colaboración de otras clases, tanto del modelo como de la vista, y además hace llamada a la método insertIntoLista(sql,

ResultSet sol, guardia_id, profesor), con el objetivo de ingresar las guardias a la lista de cada usuario como se representa en la siguiente ilustración.

```
private void createGuadiaList() throws SQLException {
    String sql = "SELECT `id`,`brigada_id`,`profesorAlFrente_id` FROM `Guardia` ORDER BY `id`";
    ResultSet sol = conexion.SELECT(sql);
    while (sol.next()) {
        int guardia_id = sol.getInt("id");
        sql = "SELECT `id` FROM `Usuario` WHERE `brigada_id` = " + sol.getInt("brigada_id");
        ResultSet sol1 = conexion.SELECT(sql);
        sql = "INSERT INTO `listaGuardia`( `usuario_id`, `guardia_id`) VALUES (?,?)";
        while (sol1.next()) {
            insertIntoLista(sql, sol1, guardia_id, -1);
        }
        insertIntoLista(sql, null, guardia_id, sol.getInt("profesorAlFrente_id"));
    }
}
```

```
private void insertIntoLista(String sql, ResultSet sol, int guardia_id, int profeso:{
    List<ObjetoData> objeto = new LinkedList<ObjetoData>();
    if (sol == null) {
        objeto.add(new IntegerData(profesor));
    } else {
        int usuario_id = sol.getInt("id");
        objeto.add(new IntegerData(usuario_id));
    }
    objeto.add(new IntegerData(guardia_id));
    conexion.EVALUAR(sql, objeto);
}
```

Ilustración 6 Ejemplo de utilización del patrón alta cohesión

Experto: Dentro de la arquitectura del *framework Symfony 2*, este patrón se muestra muy evidenciado en la capa perteneciente al modelo de datos. En la misma se encuentran las clases encargadas de interactuar directamente con la base de datos a través del ORM Doctrine, las cuales generalmente se encuentran bajo el nombre de *name_entityRepository*, presentando la responsabilidad de realizar directamente las acciones de consultas debido a que conocen los atributos necesarios para realizar dichas operaciones. En la aplicación se evidencia este patrón en las entidades Usuario.php, Guardia.php, Reporte.php, Asistencia.php, entre otras.

Controlador: En *Symfony2* todas las peticiones web son manipuladas por el controlador frontal, que es el punto de entrada de toda la aplicación en un entorno determinado. Los eventos se separan en varias acciones para que se aumente la cohesión y disminuya el acoplamiento. En el caso de la aplicación a

desarrollar se evidencia en las clases controladoras DefaultController.php, GuardiaController.php, ReporteController.php, entre otras.

Patrones de diseño GoF

Los patrones GoF se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento. Seguidamente se presenta el patrón GoF utilizado para el diseño del sistema. (Gamma, 1995)

Decorador: Este patrón plantea la utilización de una o varias plantillas globales, que guarden el código que es usual en varias plantillas del sistema, para no tener que repetirlo en cada interfaz. (Eguiluz, 2011) En la presente aplicación se utilizó una herencia de dos niveles para reutilizar el máximo código posible como se representa en la siguiente ilustración.

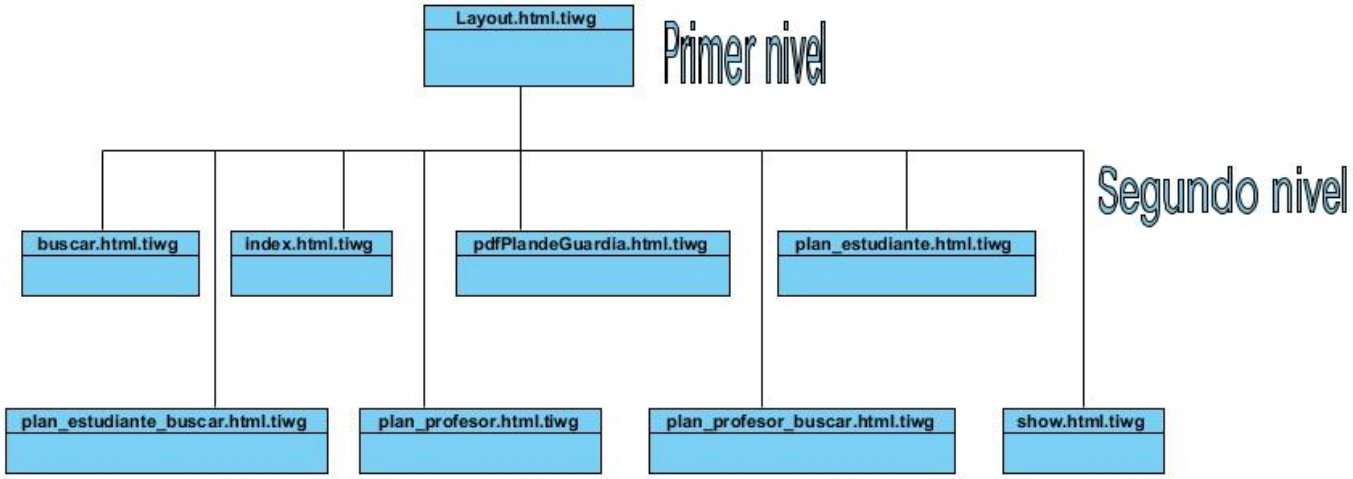


Ilustración 7 Ejemplo de utilización del patrón decorador

Estrategia: Este patrón plantea la utilización de varias estrategias para representar comportamientos distintos, utilizando una misma interfaz. Estos comportamientos pueden ser modificados e intercambiados en la aplicación, sin que afecte la lógica del sistema. En la aplicación se evidencia en el método denegarCambioAction($id) al tener en cuenta la categoría del usuario para notificarle del estado de su solicitud de intercambio de turnos como se representa en la siguiente ilustración.

```
public function denegarCambioAction($id) {


    $em = $this->getDoctrine()->getManager();
    $mensaje = $em->getRepository('SPCGOEUsuarioBundle:Mensaje')->find($id);
    $usuario_solicita = $mensaje->getUserQSolicitaCambiarGuardia();
    $mail = "";

    if ($usuario_solicita->getCategoria() == "Estudiante") {
        $mail = $mensaje->getUserQSolicitaCambiarGuardia() . '@estudiantes.uci.cu';
    } else {
        $mail = $mensaje->getUserQSolicitaCambiarGuardia() . '@uci.cu';
    }

    $mensaje->setSolicitudRevisada(TRUE);
    $mensaje->setAutorizarCambio(FALSE);


    $em->persist($mensaje);
    $em->flush();

    $mensaje1 = \Swift_Message::newInstance()
            ->setSubject('Sistema de Planificacion de la Guardia Obrero-Estudiantil')
            ->setFrom('jaromero@estudiantes.uci.cu')
            ->setTo($mail)
            ->setBody('El compañero ' . $mensaje->getUserSolicitadoParaCambiarGuardia()->getNombre() .
    $this->get('mailer')->send($mensaje1);

    return $this->redirect($this->generateUrl('portada'));
}
```

Ilustración 8 Ejemplo de utilización del patrón estrategia

3.5 Estándar de codificación

Durante el proceso de implementación de un software es considerado una buena práctica realizar la codificación del mismo siguiendo estándares que guíen este proceso. La metodología XP propone el uso de estándares de codificación, reforzando esta práctica con otras como la programación en parejas, la propiedad colectiva del código y la integración continua. De modo que el código no sea conocido por una sola persona del equipo de desarrollo, esto permite que otros miembros del equipo puedan realizar cambios en el código. (alan2793, 2013)

A continuación se muestran los principales estándares de codificación utilizados en la implementación de la propuesta de solución.

- Añadir un solo espacio después de cada delimitador (coma).
- Añadir un solo espacio alrededor de los operadores (==, and, or, y otros).
- Añadir una (coma) después de cada elemento del arreglo en un arreglo multilíneas, incluso después del último.

- El código se encuentra tabulado a través del formato que aplica la combinación de teclas ALT+SHIFT+F del NetBeans IDE.
- Los comentarios multilíneas se escriben comenzando con los caracteres "/*" y terminando con "*/", los comentarios de una sola línea comienzan con los caracteres "//".
- Usar llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.
- Declarar las propiedades de clase antes que los métodos. Utilizar paréntesis al crear instancias de clases, independientemente del número de argumentos que el constructor tenga.
- Los nombres de las clases del modelo utilizan el estándar *Upper Camel Case* y las clases controladoras y los métodos utilizan en sus nombres el estándar *Camel Case*.
- Todos los atributos de las clases del modelo están encapsulados. Son privados y el acceso a ellos se hace por medio de los métodos getAtributo y/o setAtributo.
- Las variables deben ser explícitas, aunque se pueden usar abreviaturas siempre y cuando no violen este principio.

El uso de estándares facilitará una mejor comprensión del código por los desarrolladores, durante la implementación y en futuros mantenimientos. El código mostrará una mayor limpieza, claridad y organización, elementos estos que aportan un valor agregado a la propuesta de solución.

3.6 Bundles del sistema

Los bundles son un conjunto estructurado de archivos que se encuentran en un directorio y constituyen la parte más importante del marco de trabajo *Symfony* 2. Permiten utilizar funcionalidades construidas por otras personas o las propias del programador para distribuirlas y reutilizarlas en otros proyectos. (Eguiluz, 2011)

A continuación se especifican los bundles usados en la solución y sus funcionalidades:

GuardiaBundle: Es donde se gestiona el proceso de planificación y control del SGOE.

UsuarioBundle: Se encarga de la gestión de los usuarios, mostrar la asistencia, y otros.

3.7 Tareas de ingeniería

La metodología de desarrollo XP propone dividir cada HU en Tareas de Ingeniería (TI) con el objetivo de facilitar la implementación a los programadores. Una HU se puede desglosar en varias TI para su desarrollo, donde se detallarán los pasos seguidos para obtener un resultado satisfactorio en la implementación de la funcionalidad asociada a la HU descrita por el cliente. (Ing.Joskowicz, 2008)

Las HU serán implementadas de acuerdo a la planificación realizada en las iteraciones definidas. A continuación se detallan cada una de las iteraciones efectuadas:

Primera Iteración

En esta iteración se desarrollaron las funcionalidades básicas del sistema que darán soporte a las funciones más complejas de la segunda iteración.

Tabla 17 Tiempo real por cada HU en la iteración 1

Historia de usuario	Tiempo	Tiempo real
Autenticar usuario	1	0.4
Ingresar datos del usuario	1	1
Crear brigadas del SGOE	2	1
Generar planificación del SGOE	2	2
Replanificar SGOE	1	0.5

Segunda Iteración

En esta iteración se desarrollaron las funcionalidades correspondientes al control (en cuanto a cumplimiento) del SGOE.

Tabla 18 Tiempo real por cada HU en la iteración 2

Historia de usuario	Tiempo estimado	Tiempo real
Mostrar planificación del SGOE	1	0.5
Mostrar mi planificación del SGOE	1	0.5
Crear reporte de cumplimiento del SGOE	1	1
Mostrar reporte de cumplimiento del SGOE	1	0.4
Mostrar asistencia al SGOE	1	0.3

Tercera Iteración

En esta iteración se desarrollaron las funcionalidades correspondientes a intercambiar turnos del SGOE.

Tabla 19 Tiempo real por cada HU en la iteración 3

Historia de usuario	Tiempo estimado	Tiempo real
Buscar integrantes en el SGOE para solicitar intercambio de turnos	1	1

Intercambiar turnos en el SGOE	1	1
Autorizar intercambio de turnos en el SGOE	1	0.5
Listar intercambio de turnos en el SGOE	1	0.3
Mostrar leyenda	1	0.4

Las HU redactadas en el Capítulo 2 fueron desglosadas en un total de 22 TI, en este epígrafe solo se muestran las primeras cinco TI correspondientes a las HU 1, HU 2, HU 3,HU 4 e HU 5.Las restantes se encuentran en los anexos de la investigación (ver Anexo 2).

Tabla 20 TI _Número 1

Tarea de ingeniería	
No. De la tarea: 1	No. De la HU: 1
Nombre de la terea: Autenticar usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio:17/2/2014	Fecha fin: 21/2/2014
Programador responsable: José Angel Díaz Romero	
Descripción: Implementar la plantillalogin.html.twig y el método loginAction () de la clase DefaultController.php.	

Tabla 21 TI _Número 2

Tarea de ingeniería	
No. De la tarea: 2	No. De la HU: 2
Nombre de la terea: Actualizar datos del usuario	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio:21/2/2014	Fecha fin: 28/2/2014
Programador responsable: José Angel Díaz Romero	
Descripción: Implementar los método insertarDatosUsuarioAction (), subirCSV() de la clase UsuarioController.php.	

Tabla 22 TI _Número 3

Tarea de ingeniería	
No. De la tarea: 3	No. De la HU: 3
Nombre de la terea: Implementar proyecto Java para crear las brigadas	

Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha inicio:1/4/2014	Fecha fin:22/4/2014
Programador responsable: José Angel Díaz Romero	
Descripción: Crear un algoritmo empleando el lenguaje de programación Java para planificar el SG. Implementar el método main () de la clase CrearBrigada, implementar los métodos getId () de la clase Estudiante. Implementar el método InsertarBrigadas (LinkedList<Brigada> lista) para insertar en la base de datos.	

Tabla 23 TI _Número 4

Tarea de ingeniería	
No. De la tarea:4	No. De la HU: 4
Nombre de la terea: Implementar proyecto Java para planificar el SGOE	
Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha inicio:1/4/2014	Fecha fin:22/4/2014
Programador responsable: José Angel Díaz Romero	
Descripción: Crear un algoritmo empleando el lenguaje de programación Java para planificar el SG. Implementar el método main () de la clase GenerarGuardia, implementar los métodos getId () de la clase Profesor. Implementar el método AlmacenarPlanificacion (LinkedList<Guardia> lista) para insertar en la base de datos y los métodos siguienteDia (), Planificar () de la clase Planificador. Implementar la clase Conexion para acceder a la base de datos y hacer modificaciones, la clase DateData, la clase IntegerData, la clase ObjectData. Implementar los métodos getId (), marcar (posición) y libre (posición). Implementar los métodos getBrigada (), getProfesor (), getFecha (), getTurno () y getArea () de la clase Guardia.	

Tabla 24 TI _Número 5

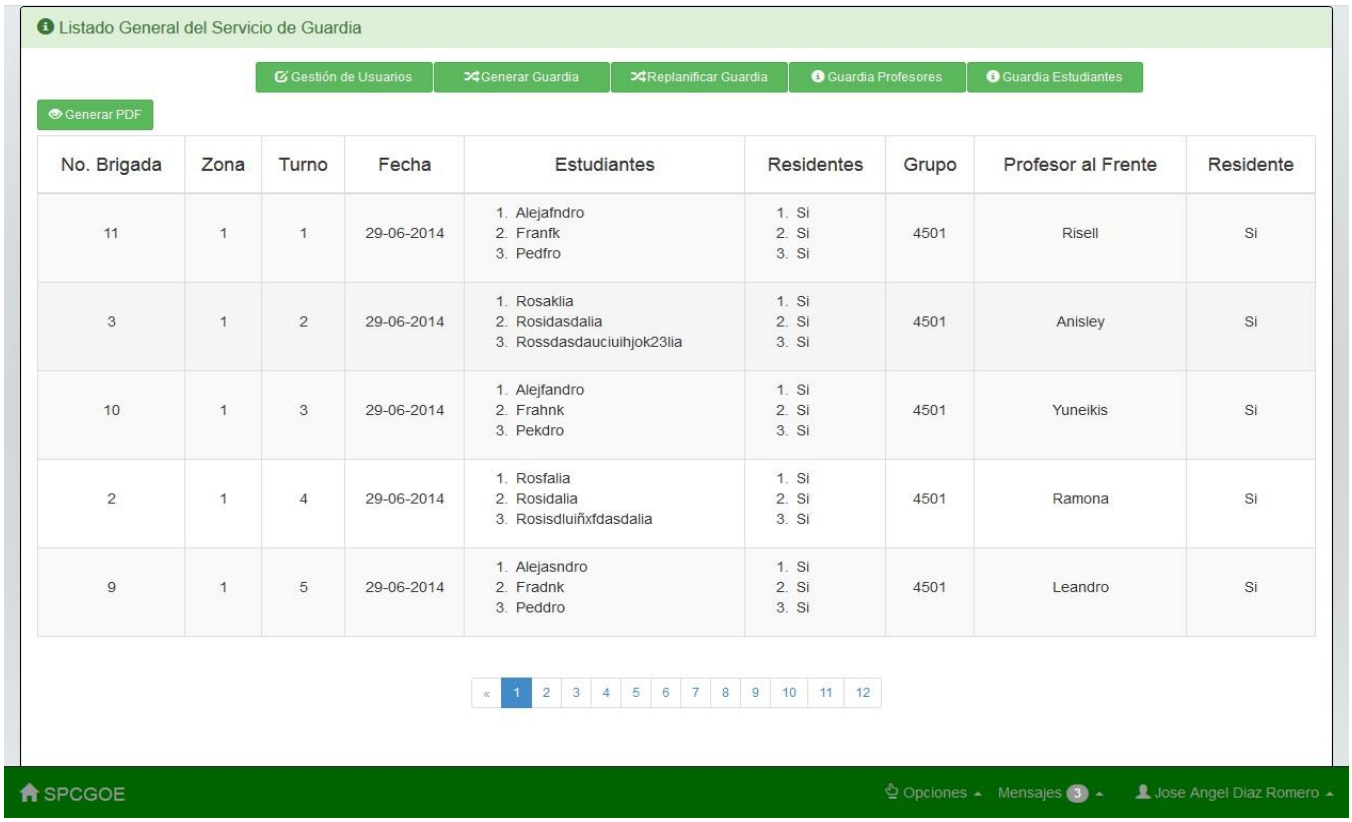
Tarea de ingeniería	
No. De la tarea: 5	No. De la HU: 5
Nombre de la terea: Implementar proyecto Java para replanificar el SGOE	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio:1/4/2014	Fecha fin:22/4/2014
Programador responsable: José Angel Díaz Romero	

Descripción: Crear un algoritmo empleando el lenguaje de programación Java para replanificar el SG. Implementar el método main () de la clase GenerarGuardia, implementar los métodos getId () de la clase Profesor. Implementar el método AlmacenarPlanificacion (LinkedList<Guardia> lista) para insertar en la base de datos y los métodos siguienteDia (), Planificar () de la clase Planificador. Implementar la clase Conexion para acceder a la base de datos y hacer modificaciones, la clase DateData, la clase IntegerData, la clase ObjectData. Implementar los métodos getId (), marcar (posición) y libre (posición). Implementar los métodos getBrigada (), getProfesor (), getFecha (), getTurno () y getArea () de la clase Guardia.

3.8 Análisis del funcionamiento del sistema

El sistema está compuesto por 4 módulos (Gestionar guardia, Gestionar usuario, Tomar asistencia, Realizar intercambio de turno) cumpliendo con la propuesta de los procesos automatizables definidos en el Capítulo 2 e implementados en el epígrafe anterior mediante el cumplimiento de las tareas de la ingeniería. A continuación se explica el funcionamiento de los módulos mencionados anteriormente.

Modulo Gestionar guardia: Gestiona toda la información del SGOE, permite planificar y replanificar la guardia. Además muestra un listado de las guardias planificadas en el sistema y posibilita conocer las guardias de la persona autenticada en el sistema. A continuación se muestran las vistas para dichas funcionalidades, Ilustraciones 9, 10 y 11.



No. Brigada	Zona	Turno	Fecha	Estudiantes	Residentes	Grupo	Profesor al Frente	Residente
11	1	1	29-06-2014	1. Alejafndro 2. Franfk 3. Pedfro	1. Si 2. Si 3. Si	4501	Risell	Si
3	1	2	29-06-2014	1. Rosaklia 2. Rosidasdalia 3. Rossdasdauciuihjok23lia	1. Si 2. Si 3. Si	4501	Anisley	Si
10	1	3	29-06-2014	1. Alejfandro 2. Frahnk 3. Pekdro	1. Si 2. Si 3. Si	4501	Yuneikis	Si
2	1	4	29-06-2014	1. Rosfalia 2. Rosidalia 3. Rosisdluiñxfdasdalia	1. Si 2. Si 3. Si	4501	Ramona	Si
9	1	5	29-06-2014	1. Alejasndro 2. Fradnk 3. Peddro	1. Si 2. Si 3. Si	4501	Leandro	Si

Ilustración 9 Planificación del SGOE

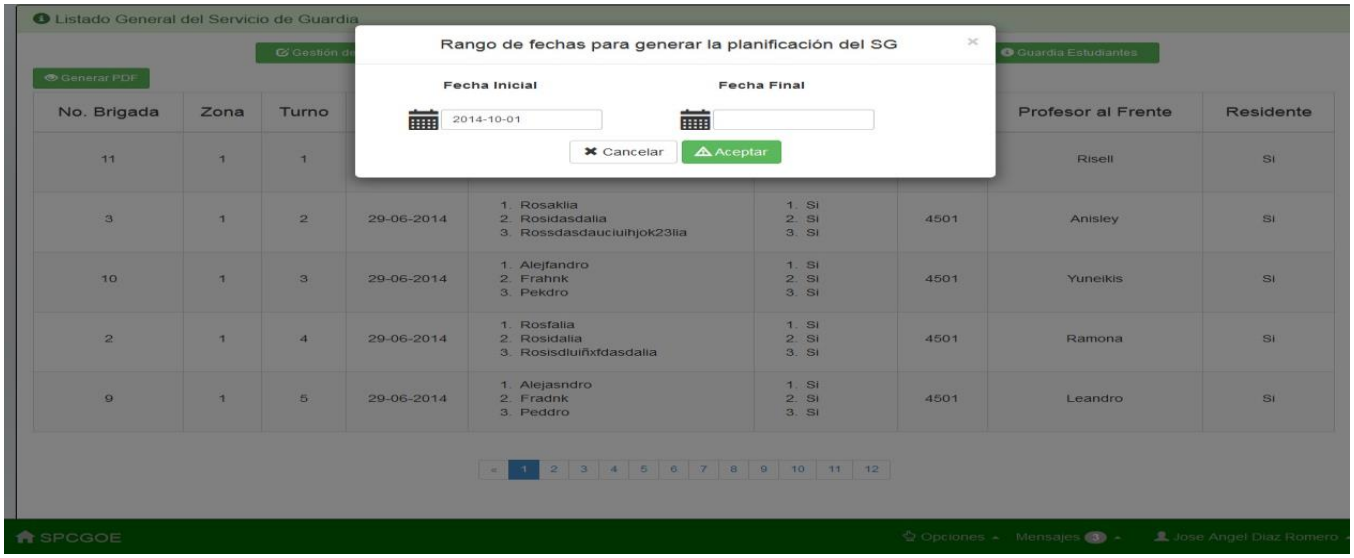


Ilustración 10 Vista planificar SGOE

Modulo Gestionar usuario: Gestiona toda la información de los usuarios que se encuentran en el sistema, permite modificar los datos (disponibilidad, sexo, grupo, categoría y otros) de los mismos, además posibilita adicionar un usuario nuevo al sistema. A continuación se muestran las vistas para dichas funcionalidades, Ilustraciones 12, 13, 14 y 15.

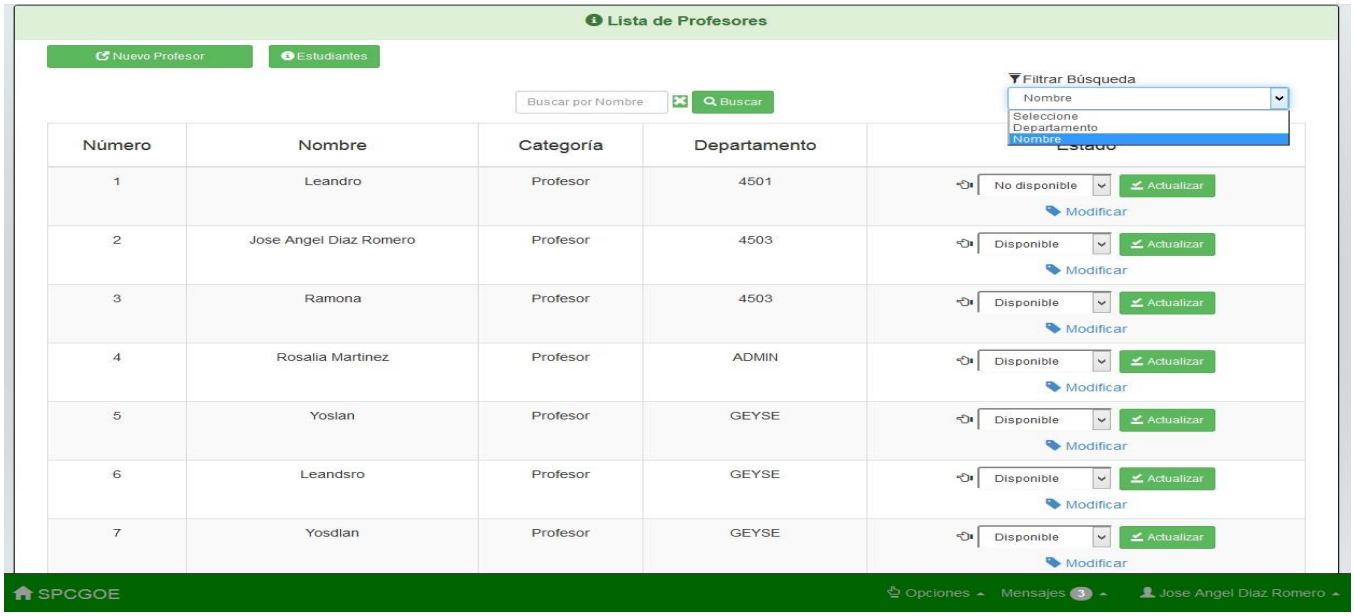


Ilustración 11 Lista de profesores

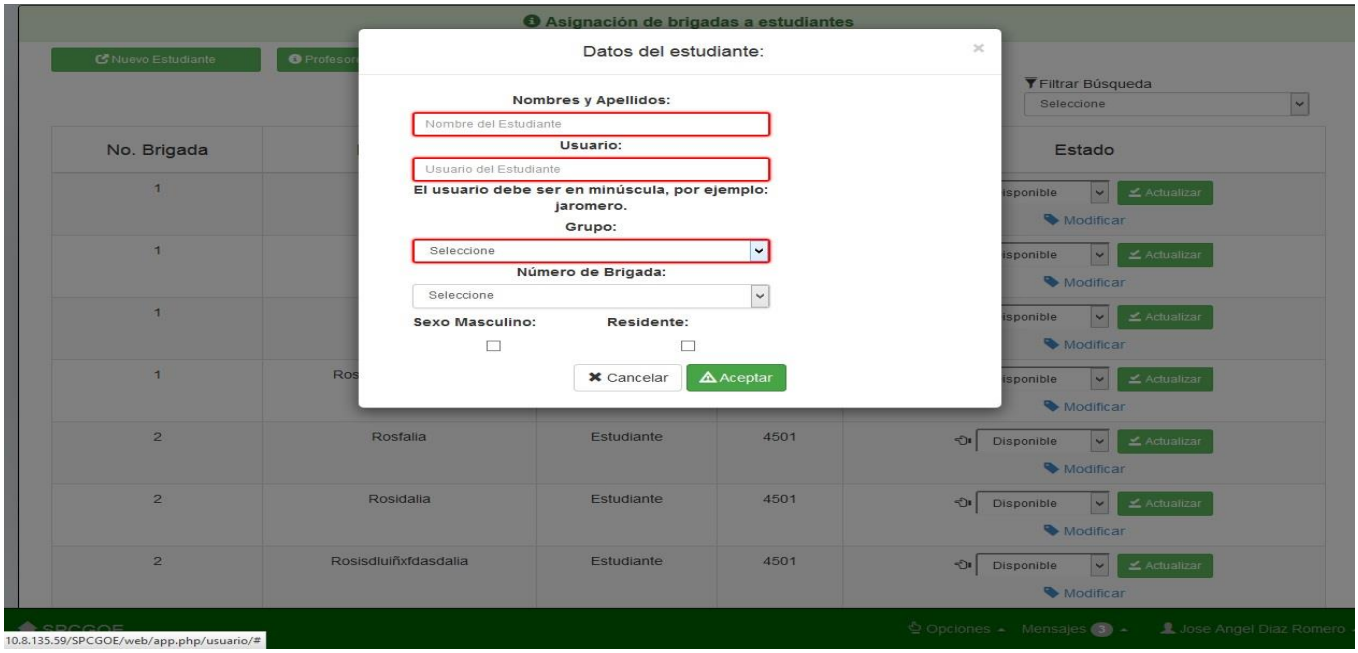


Ilustración 12 Nuevo estudiante

Modulo Tomar asistencia: Gestiona toda la información referente a la asistencia de cada usuario al SGOE. Permite realizar reportes diarios de la asistencia en una fecha determinada, además de mostrar, registrar y modificar la asistencia creada. A continuación se muestran las vistas para dichas funcionalidades, Ilustraciones 16 y 17.

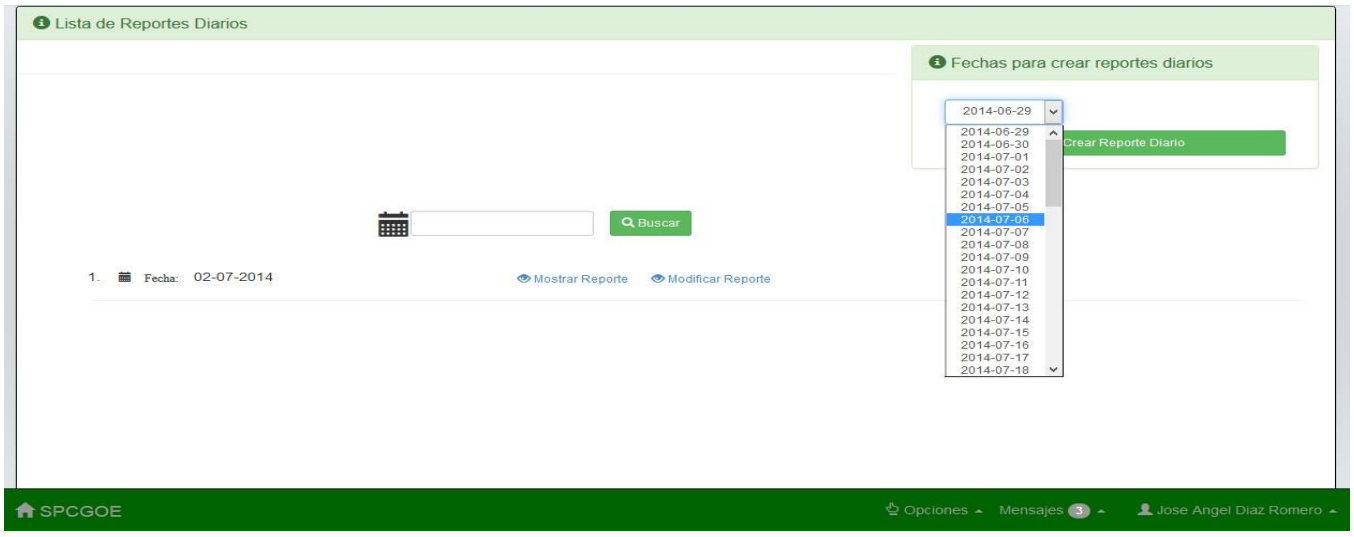


Ilustración 13 Lista de reportes diarios

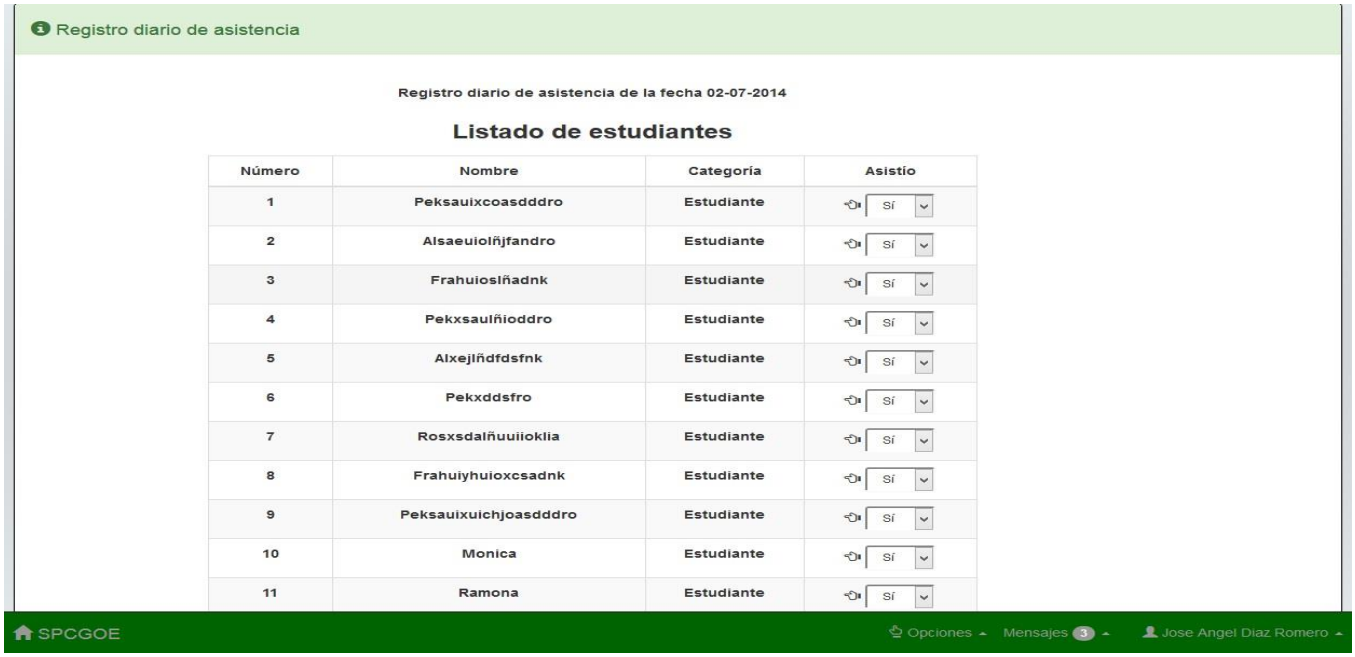


Ilustración 14 Registro diario de asistencia

Modulo Realizar intercambio de turno: Gestiona toda la información de los cambios de turnos realizados en el SGOE. Permite crear mensajes de solicitud de intercambio de turnos que serán enviados a los usuarios solicitados para intercambiar turnos. Además posibilita crear el cambio una vez aceptada la solicitud de cambio. A continuación se muestran las vistas para dichas funcionalidades, Ilustraciones 18 y 19.

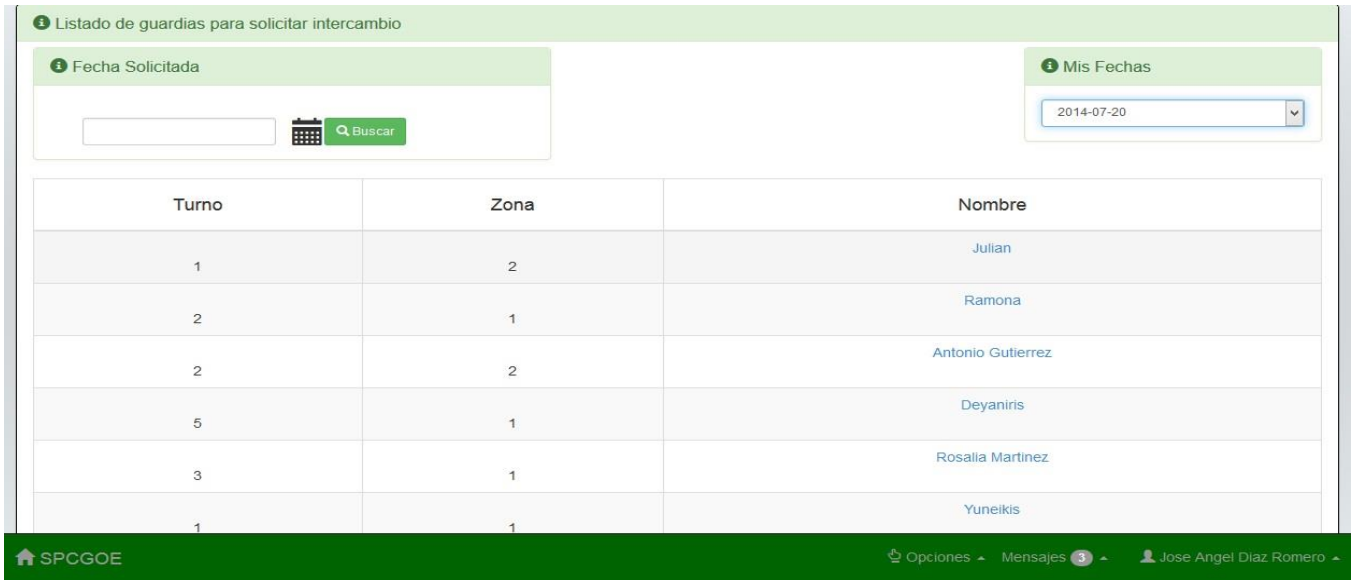


Ilustración 15 Listado de guardias para solicitar intercambio

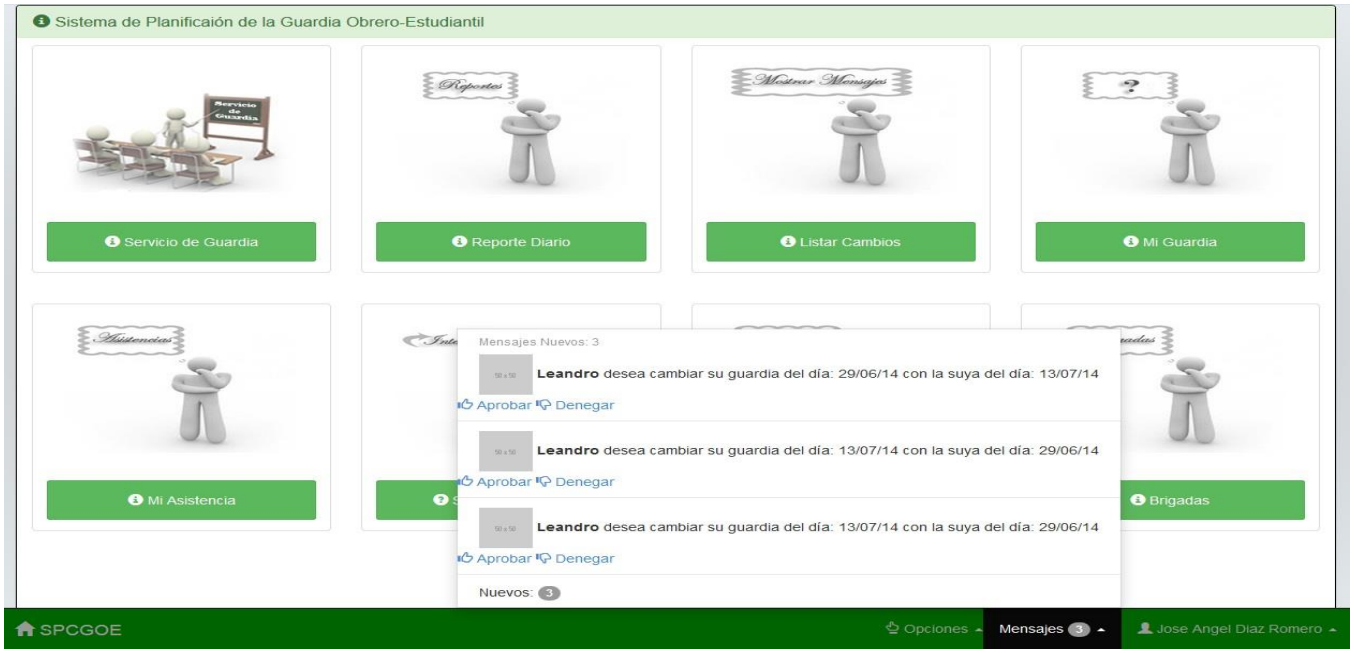


Ilustración 16 Gestión de mensajes

3.9 Pruebas

La Prueba es la disciplina del proceso de ingeniería de software cuyo propósito es integrar y probar el sistema. Elemento crítico para la garantía de la calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación. Su objetivo fundamental es descubrir diferentes clases de errores con la menor cantidad de tiempo y de esfuerzo. Dentro de los procedimientos para la realización de las pruebas al software se encuentran las pruebas unitarias y las pruebas de aceptación. (Gutierrez, et al., 2010)

3.9.2 Pruebas unitarias

Symfony 2 ha optado por utilizar la librería PHPUnit, de esta forma, los *tests* (pruebas) unitarios de *Symfony* 2 combinan la potencia de PHPUnit con las utilidades y facilidades proporcionadas por el *framework* de desarrollo. Por convención, cada *test* unitario se define en una clase cuyo nombre acaba en Test encontrándose dentro del directorio Tests/ del *bundle*. *PHPUnit* utiliza *assertions* (aserciones) para verificar que el comportamiento de una unidad de código es el esperado. Cuando se produce un error, *PHPUnit* muestra el texto "*failures*" (fallo) como resumen de la ejecución. Antes muestra el listado de todos los tests que han fallado, indicando para cada error la clase y método erróneos. El mensaje "*ok*" (bien) indica que todos los *tests* se han ejecutado correctamente. (Eguiluz, 2011)

Estas pruebas fueron realizadas a las clases Guardia.php, Mensaje.php y Asistencia.php del Modelo de datos como se representa en las siguientes ilustraciones.

```
PHPUnit 4.1.0 by Sebastian Bergmann.

Configuration read from D:\TESIS\APLICACION\WAMP\wamp\www\MITESIS\app\phpunit.xml.dist

ESC[41;37mFESC[0m

Time: 414 ms, Memory: 6.75Mb

There was 1 failure:

1) SPCGOE\GuardiaBundle\Tests\PlanificarGuardia\MensajeTest::testsetvalidarMensajeTest
Mensaje
Failed asserting that 0 matches expected 9.

D:\TESIS\APLICACION\WAMP\wamp\www\MITESIS\src\SPCGOE\GuardiaBundle\Tests\PlanificarGuardia\GuardiaTest_2.php:21
ESC[37;41m                                        ESC[0m
ESC[37;41mFAILURES!                               ESC[0m
ESC[37;41mTests: 1, Assertions: 1, Failures: 1.ESC[0m
```

Entidad: Mensaje

Ilustración 17 Resultado PU a la entidad Mensaje

```
PHPUnit 4.1.0 by Sebastian Bergmann.

Configuration read from D:\TESIS\APLICACION\WAMP\wamp\www\MITESIS\app\phpunit.xml.dist

ESC[41;37mFESC[0m

Time: 390 ms, Memory: 6.75Mb

There was 1 failure:

1) SPCGOE\GuardiaBundle\Tests\PlanificarGuardia\GuardiaTest::testsetvalidarGuardiaTest
Guardia
Failed asserting that 0 matches expected 5.

D:\TESIS\APLICACION\WAMP\wamp\www\MITESIS\src\SPCGOE\GuardiaBundle\Tests\PlanificarGuardia\GuardiaTest_5.php:21
ESC[37;41m                                        ESC[0m
ESC[37;41mFAILURES!                               ESC[0m
ESC[37;41mTests: 1. Assertions: 1. Failures: 1.ESC[0m
```

Entidad: Guardia

Ilustración 18 Resultado PU a la entidad Guardia

```
PHPUnit 4.1.0 by Sebastian Bergmann.

Configuration read from D:\TESIS\APLICACION\WAMP\wamp\www\MITESIS\app\phpunit.xml.dist

ESC[41;37mFESC[0m

Time: 400 ms, Memory: 6.75Mb

There was 1 failure:

1) SPCGOE\GuardiaBundle\Tests\PlanificarGuardia\AsistenciaTest::testsetvalidarAsistenciaTest
Asistencia
Failed asserting that 0 matches expected 5.

D:\TESIS\APLICACION\WAMP\wamp\www\MITESIS\src\SPCGOE\GuardiaBundle\Tests\PlanificarGuardia\GuardiaTest_4.php:21
ESC[37;41m                                        ESC[0m
ESC[37;41mFAILURES!                               ESC[0m
ESC[37;41mTests: 1, Assertions: 1, Failures: 1.ESC[0m
```

Entidad: Asistencia

Ilustración 19 Resultado PU a la entidad asistencia

Fueron realizadas tres iteraciones de las pruebas Unitarias, en las cuales se detectaron los siguientes errores:

1era iteración:

- Algunas funcionalidades no validaban los datos de entrada.
- No se validaba que todos los campos deben ser llenados, es decir que no se dejen campos vacíos.

2 da iteración:

- No se validaba que todos los campos deben ser llenados, es decir que no se dejen campos vacíos.

Estos errores fueron corregidos de manera inmediata y en la tercera iteración no se detectaron errores, por tanto las funcionalidades probadas funcionan correctamente.

3.9.3 Pruebas de aceptación

Las pruebas de aceptación son las especificaciones para el comportamiento deseado y la funcionalidad de un sistema. Describen, por una HU dada, cómo el sistema se encarga de ciertas condiciones e insumos y con qué tipo de resultados. Los clientes junto a un miembro del equipo de desarrollo son los responsables de verificar que los resultados de estas pruebas sean los correctos para así tomar decisiones acerca de las mismas. Una HU no se puede considerar terminada hasta que no pase las pruebas de aceptación. Es recomendable publicar los resultados de las pruebas de aceptación, para que todo el equipo de desarrollo esté al tanto de esta información. (Gutierrez, et al., 2010)

A continuación se muestran algunos casos de prueba de aceptación. El resto de los casos de prueba de aceptación se encuentran en Anexo 3.

Tabla 25 PA Autenticar Usuario

Pruebas de aceptación	
Código:1	No. HU: 1
Nombre: Autenticar Usuario	
Descripción: Prueba para la funcionalidad que permite al usuario autenticarse, asegurando la seguridad del sistema.	
Condiciones de ejecución: Verificar que el usuario esté previamente autenticado en el sistema.	
Pasos de ejecución: El usuario ingresa su usuario y contraseña, el sistema valida la veracidad de los datos y seguidamente se muestra la interfaz principal del sistema.	
Resultado esperado: Que el usuario pueda autenticarse satisfactoriamente.	

Evaluación de la prueba: Prueba satisfactoria.

Tabla 26 PA Actualizar Datos del Usuario

Pruebas de aceptación	
Código: 2	No. HU: 1
Nombre: Actualizar Datos del Usuario	
Descripción: Prueba para la funcionalidad que permite al usuario actualizar los datos de los usuarios.	
Condiciones de ejecución: El usuario debe estar autenticado con el rol de Súper-Administrador.	
Pasos de ejecución: El usuario selecciona la opción “Servicio de Guardia” que se encuentra en el menú principal. Luego se acciona sobre el botón “examinar” y se le muestra un diálogo con el cual puede seleccionar un archivo en formato csv que contiene los datos de los usuarios, si el formato no coincide se muestra el mensaje “Formato no válido, solo se admite el formato csv”. Se le brinda la posibilidad de actualizar los datos del usuario: Nombre y apellidos, usuario, departamento, categoría docente, residencia (si el usuario reside o no en la Universidad).	
Resultado esperado: Se registran en la base de datos del sistema los datos correspondientes a cada usuario.	
Evaluación de la prueba: Satisfactoria.	

Tabla 27 PA Crear Brigadas del SG

Pruebas de aceptación	
Código:3	No. HU: 2
Nombre: Crear Brigadas del SG	
Descripción: Prueba para la funcionalidad que permite al administrador crear las brigadas de usuarios del SG.	
Condiciones de ejecución: El usuario debe estar autenticado con el rol de Administrador.	
Pasos de ejecución: El administrador selecciona la opción “Servicio de Guardia” que se encuentra en el menú principal. Luego selecciona la opción “Gestión de usuario” y el sistema muestra una lista con los datos de todos los usuarios creados en la aplicación. El administrador selecciona el estado de disponibilidad (disponible, no disponible) de un usuario y selecciona “Actualizar”. El sistema actualiza el estado del usuario y muestra el estado de disponibilidad actual. Finalmente el administrador selecciona la opción “Asignar Brigadas” y se crean todas las brigadas que compondrán el SG. El sistema muestra las brigadas creadas.	
Resultado esperado: El sistema crean las brigadas que compondrán el SG.	
Evaluación de la prueba: Satisfactoria.	

3.9.4 Resultado de las pruebas

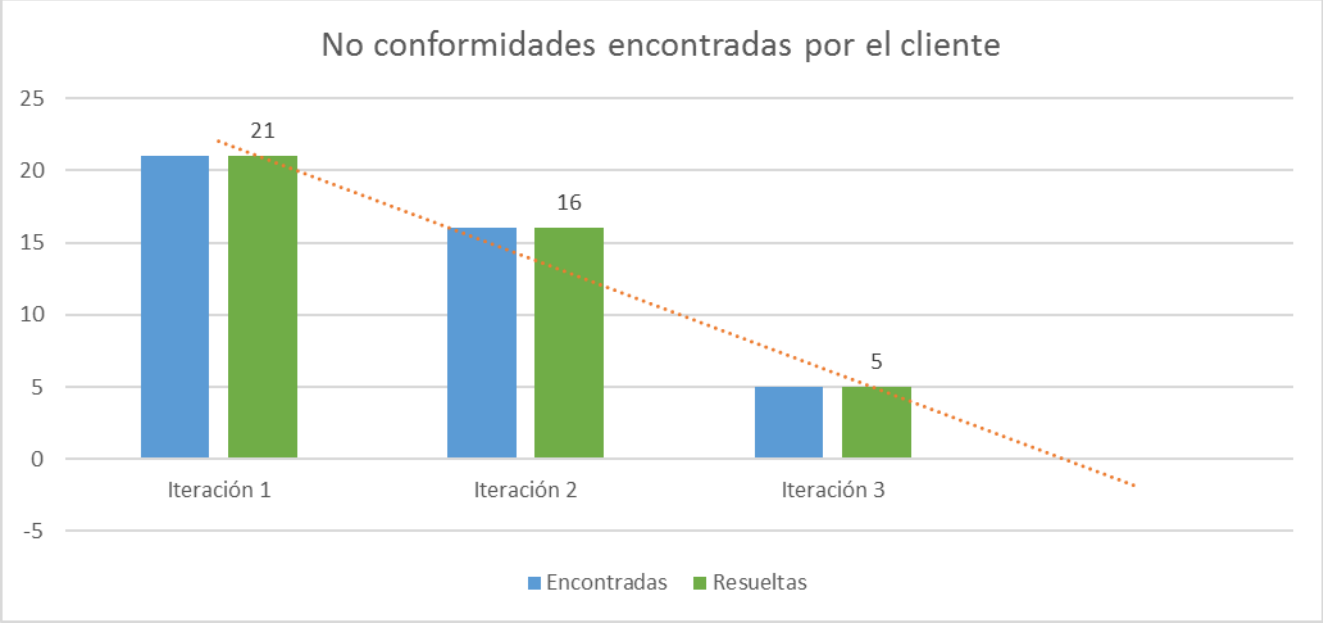
Se elaboraron 15 casos de pruebas de aceptación, uno por cada funcionalidad o HU y 1 caso de prueba de aceptación para el RNF de rendimiento (pruebas de carga y estrés).

Luego de ejecutadas las pruebas de aceptación, se detectaron algunas no conformidades como errores ortográficos, problemas de interfaz visual y errores en la validación de algunos campos. Las no conformidades encontradas por cada iteración fueron resueltas antes de pasar a la siguiente iteración. La Tabla 19 muestra el resultado de las pruebas en cada una de las iteraciones.

Tabla 28 Tipo de errores encontrados en las PA

Tipo de Error	Ortográfico	Funcionalidades	Validación	Total
Iteración1	16	2	3	**21**
Iteración2	10	4	2	**16**
Iteración3	0	3	2	**5**

Tabla 29 Análisis de los resultados de las PA



Se utilizó la herramienta JMETER para realizar las pruebas de carga y estrés, simulando 150 usuarios realizando 4 peticiones al servidor simultáneamente, el tiempo de respuesta promedio fue de 1417 a 4717 mili-segundos y el tiempo de respuesta máximo fue de 8820 mili-segundos. El resultado de la prueba resulto satisfactorio.

A continuación se muestra una ilustración que contiene los resultados de la prueba realizada:

Summary Report

Nombre: Summary Report

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

Label	# Muestras	Media	Mín	Máx	Std. Dev.	% Error	Rendimiento	Kb/sec	Avg. Bytes
/SPCGOE/web/app.php/guardia/	150	4717	374	8820	2729,36	0,00%	5,1/sec	57,75	11484,0
/SPCGOE/web/app.php/mensaje/	150	1823	346	4061	1014,11	0,00%	4,2/sec	47,66	11484,0
/SPCGOE/web/app.php/user_asistencia/mia	150	1697	132	3697	797,89	0,00%	4,0/sec	44,39	11484,0
/SPCGOE/web/app.php/usuario_guardia/buscar	150	1417	336	3771	880,01	0,00%	4,0/sec	44,82	11484,0
TOTAL	600	2413	132	8820	2064,50	0,00%	14,1/sec	158,68	11484,0

Ilustración 20 Resultado de la prueba de carga y estrés

3.10 Conclusiones del capítulo

Durante el progreso del capítulo que concluye se generaron artefactos necesarios para llevar a cabo la implementación de la propuesta de solución, como es el caso de las tarjetas CRC, las cuales posibilitaron identificar las entidades presentes en la propuesta de solución y las relaciones entre ellas. Se evidenció la importancia que requiere realizar un correcto diseño del software basado en buenas prácticas como es el uso de patrones arquitectónicos y de diseño. Los estándares y estilos de codificación utilizados permitieron establecer un mejor entendimiento del código, facilitando el posterior mantenimiento y modificación del mismo. Las tareas de ingenierías dieron paso a una correcta implementación de las funcionalidades descritas en cada HU. Las pruebas unitarias y de aceptación arrojaron errores que fueron corregidos en cada iteración, lo que permitió la obtención de un software con la calidad exigida por el cliente.

CONCLUSIONES GENERALES

Con el desarrollo de la presente investigación se dio respuesta a los objetivos propuestos, arribando a las siguientes conclusiones:

- El estudio realizado en base a la bibliografía de los sistemas similares existentes y el diagnóstico del proceso actual de planificación y control del SGOE en la Facultad 4 de la UCI posibilitaron: Elaborar los referentes teóricos-metodológicos fundamentales relacionados con la gestión de procesos universitarios de manera general y con la planificación y control de servicios de guardia mediante aplicaciones. Además de identificar los requerimientos con que debe cumplir la propuesta de solución
- La investigación sobre la selección de las herramientas y tecnologías existentes para guiar el proceso de desarrollo del sistema informático y la confección de los artefactos que propone la metodología de desarrollo de software seleccionada permitieron realizar la implementación de la propuesta de solución planteada.
- La aplicación de pruebas al sistema, tales como pruebas unitarias y pruebas de aceptación, permitieron validar que la herramienta desarrollada cumple con los requisitos funcionales determinados por el cliente.

RECOMENDACIONES

Tomando como base la investigación realizada y el análisis de los resultados obtenidos se recomienda:

- Optimizar el algoritmo para planificar el SGOE y de esta forma lograr una mejor distribución de la planificación.
- Agregar funcionalidades que permitan planificar las cuartelerías en la residencia de la Facultad 4 de la UCI.

REFERENCIAS BIBLIOGRÁFICAS

1. **alan2793. 2013.** Entorno de Desarrollo Integrado(IDE). *Entorno de Desarrollo Integrado(IDE).* [En línea] 2013. [Citado el: 2014 de 02 de 03.] http://alanss18.wordpress.com/2013/01/25/entorno-de-desarrollo-integradoide/. http://alanss18.wordpress.com/2013/01/25/entorno-de-desarrollo-integradoide/.
2. **Alonso, J. 2006.***¿Qué es un Framework?*.
3. **Alvarez, Miguel Angel. 2009.** DesarrolloWeb. [En línea]. http://www.desarrolloweb.com/articulos/codeigniter.html.
4. **—. 2001.** DesarrolloWeb. [En línea] 2001. http://www.desarrolloweb.com/articulos/541.php.
5. **Alvarez, Sara. 2007.** Desarrollo Web. [En línea].
6. **Angel, Alvarez Miguel. 2003.** DesarrolloWeb. [En línea] . http://www.desarrolloweb.com/articulos/1325.php.
7. **Antonio, Juan. 2000.** TutorJava. *TutorJava.* [En línea] 17 de Septiembre de 2000. [Citado el: 2 de Abril de 2014.] ftp://10.22/documentacion/Programacion/Java/Tutorial%20de%20Java%20en%20Espanol/index.html.
8. **Bakken, Stig Sæther, y otros. 2001.***Manual de PHP.* [ed.] Rafael Martínez.
9. **Breeding, Marshall. 2012.***Tendencias actuales y futuras en tecnologías de la información para unidades de información.* Mexico : s.n.,.
10. **Brittain, Nason y Darwin, Ian F. 2008.***Phyton: For Unix and Linux System Administration.* s.l. : O'Reilly, 2008. ISBN: 978-0-596-51582-9.
11. **Calero Solís, Manuel. 2003.***Una explicación de la programación extrema (XP)*.
12. **Campo, Luis Llorente. 2010.***Instalación y configuración del servidor web Apache.* Universidad de León: España : s.n.
13. **Ciberaula.com. 2008.** Introducción a Apache. [En línea]. http://linux.ciberaula.com/articulo/linux_apache_intro/..

14. **CORPORATION©, ORACLE. 2014.** Bienvenido a NetBeans y www.netbeans.org, Portal del IDE Java de Código Abierto. [En línea] 2014. [Citado el: 3 de Marzo de 2014.] https://netbeans.org/index_es.html..

15. **Dan, Michelangelo Van. 2013.** Slideshare. [En línea] 25 de 4 de 2013. [Citado el: 20 de 1 de 2014.] http://www.slideshare.net/DragonBe/quality-assurance-for-php-projects-with-phpstorm.

16. **Debrauwer, Laurent. 2012.***Patrones de diseño para C#.* Barcelona : Editions ENI ISBN: 978-2-7460-7260-2.

17. **Eguiluz, Javier. 2009.***CSS Avanzado*.

18. **—. 2011.***Desarrollo web ágil con Symfony2.* Primera Edición.

19. **Eguiluz, Javier Perez. 2010.***Introducción a JavaScript*.

20. **ElWebMaster. 2009.** [En línea] 2009. http://www.elwebmaster.com/articulos/frameworks-php-recomendados-guia-para-principiantes.

21. **Englander, Robert. 2002.***Java and SOAP.* May 2002. s.l. : O'Reilly, 2002. ISBN: 0-596-00175-4.

22. **Fernández Escribano, Gerardo. 2002.***Introducción a Extreme Programming*.

23. **Fowler, Martin. 2000.***Metodologías Ágiles: eXtreme Programming*.

24. **Fundation, The Apache Software. 2012.***Apache License and Distribution*.

25. **Gamma, E. 1995.***Design Patterns: Elements of Reusable Object-Oriented* Software.

26. **Gauchat, Juan Diego. 2012.***El gran libro de HTML5, CSS3 y Javascript.* s.l. : S.l: Marcombo, 2012. ISBN 8426717829.

27. **Gerrero, R. M. 2013.** "Sobre PostgresSQ". [En línea] 2013. [Citado el: 22 de 12 de 2013.] http://www.postgressql.org.es/sobre_postgressql.

28. **Gonzales, Mabel. 2011.** Monografias. [En línea]. http://www.monografias.com/trabajos14/sqlserver/sqlserver.shtml.

29. **González Barbone, Víctor A. 2007.***XP: Extreme Programming*.

30. **Gutierrez, J J y otros, y. 2010.** PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA. [En línea] 2010. [Citado el: 20 de Marzo de 2014.]

31. **IBM. 2009.** Rational Rose. [En línea] 2009. http://www-03.ibm.com/software/products/es/enterprise/.

32. **Ing.Joskowicz, Jose. 2008.***Reglas y Prácticas en eXtreme Programming.* España : s.n.

33. **Ivar Jacobson, Grady Booch, James Rumbaugh. 2000.***El Proceso Unificado de Desarrollo de* Software*.* Madrid : Pearson Education.

34. **JDBC. 2013.***Conectividad de java con bases de datos. Java Database Connectivity. JDBC.* 2013.

35. **Joskowicz, José. 2008.***XP1*.

36. **Kniberg, Henrik. 2007.***Scrum and XP from the Trenches.* Version 2.1.

37. **Lancker, Luc Van. 2009.***XHTML y CSS - Los nuevos estandares del codigo fuente.* Ediciones ENI. 2009. ISBN 9782746047426.

38. **León, Eduardo. 2000.***Tutorial Visual Paradigmfor UML*.

39. **LTD, BUILTWITH® PTY. 2014.** Twitter Bootstrap Usage Statistics. [En línea] Marzo de 2014. [Citado el: 5 de Marzo de 2014.] http://trends.builtwith.com/docinfo/Twitter-Bootstr.

40. **Luca, Damián De. 2011.***HTML5: entienda el cambio, aproveche su potencial.* s.l. : USERSHOP, 2011. ISBN 9789871773794.

41. **Martins, Miguel Angel González. 2013.***Los lenguajes de perogramación actuales.* Universidad de Alcalá de Henares : s.n.

42. **Mikoluk, Kasia. 2013.** MySQL vs PostgreSQL: Por Qué MySQL es Superior A PostgreSQL. *MySQL vs PostgreSQL.* [En línea] 2013. [Citado el: 5 de Abril de 2014.] https://www.udemy.com/blog/mysql-vs-postgresql-por-que-mysql-es-superior-a-postgresql/.

43. **Mora, Sergio Lujan. 2002.***Programación de aplicaciones web: historia, principios básicos y clientes web.* s.l. : Editorial Club Universitario, 2002. ISBN 9788484542063.

44. **Muñoz, Vicente Eslava. 2008.** Conceptos avanzados.

45. **Nagao, Fabio Zendi. 2010.***Viweb.es (jquery)*.

46. **Nieto, Tito Fernando Ale. 2010.** [En línea] 2010. http://www.monografias.com/trabajos89/introduccion-al-html-5/introduccion-al-html-5.shtml.

47. **Packo. 2012.** [En línea] 2012. http://packo.wikispaces.com/Caracteristicas+de+MYSQL.

48. **Peñalver, Gladys Romero García de la Puente, Marsi, Sergio Jesús y Meneses Abad, Abel. 2011.***SXP,metodología de desarrollo de* software *SXP,*software *development methodology.* s.l. : Serie Científica de la Universidad de las Ciencias Informáticas.

49. **PHP.net. 2010.** php.net. [En línea] 2010. http://www.php.net/.

50. **Piwen, Ariel Erlijman y Fros, Alejandro Goyén. 2001.***Problemas y soluciones en la implemantación de Extreme Programing.* Uruguay : s.n., 2001.

51. **PostgreSQL. 2012.** PostgreSQL. [En línea] 2012. http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html.

52. **Pressman, Roger S. 2010.**Software *Enginering A Practitioner's Approach.* Seventh Edition. 2010.

53. **SOLEX. 2010.** SOLEX. [En línea] 2010. [Citado el: 2 de 3 de 2014.] http://www.solex.cl/.

54. **Symfony.es. 2010.** Symfony.es. [En línea] 2010. http://symfony.es/que-es-symfony.

55. **Technologies, Z. 2009.** Programmer's Reference Guide, Zend Framework. [En línea] 2009. [Citado el: 8 de Marzo de 2014.] http://framework.zend.com/manual/en/index.html.

56. **Tisdall, James. 2001.***Beginning Perl for Bioinformatics.* [ed.] O'Reilly. First Edition October 2001. SBN: 0-596-00080-4,.

57. **visual-paradigm. 2013.** visual-paradigm. [En línea] 2013. http://www.visual-paradigm.com/.

58. **W3C. 2014.** W3C Oficina Española. [En línea] 4 de Febrero de 2014. http://www.w3c.es/Divulgacion/a-z/#h..

59. **Waite, Mitchell. 2001.***Data structures and algorithms in Java.* 2001. ISBN: 1571690956.

60. **Wheeler, Veronika. 2008.***Linux en Espoñol*.

GLOSARIO DE TÉRMINOS

Algoritmo: Es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema.

APIs: Conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación.

Framework: Es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación.

Metodologías: Conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software.

Open Source: Del español código abierto se refiere a los sistemas de software no propietario.

SGBD: Es un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que lo utilizan.