Universidad de las Ciencias Informáticas
Facultad 3

Sistema de Gestión de Información para la Atención a la Población en la Facultad 3.

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

Autores:
Carlos Javier Pérez de la Cruz
Roberto Jesús Rielo Villadóniga

Tutora:
Ing. Mailen Edith Escobar Pompa

Co-Tutoras:
MsC. Marieta Peña Abreu
MsC. Reina Estrada Nelson

La Habana, 23 de junio del 2015
¨Año 57 de la Revolución¨

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ _____

Roberto Jesús Rielo Villadóniga — Firma del Autor

Carlos Javier Pérez de la Cruz — Firma del Autor

_____

Ing. Mailen Edith Escobar Pompa

Firma del Tutor

_____ _____

MsC. Reina Victoria Estrada Nelson — Firma del Co-Tutor

MsC. Marieta Peña Abreu — Firma del Co-Tutor

"No te preguntes qué puede hacer tu país por ti, pregúntate que puedes hacer tú por tu país"

John Fitzgerald Kennedy

DEDICATORIA

De Carlos

A mis padres, porque a ellos les debo el estar hoy aquí realizando este sueño, a mi hermano por siempre estar a mi lado, a mi familia por el gran apoyo que me han dado durante toda la vida, a todas las personas que de una forma u otra han contribuido a este logro.

De Roberto

Esta es la realización de un sueño, que aunque hoy lo observes desde el cielo va dedicado a ti abuelo. A ti te debo mi inspiración y los deseos de ser más grande cada día, de ser mejor persona. Sé que has sido la fuerza que me apoya en cada paso y es por ti que aún tengo un largo camino por recorrer con grandes metas que cumplir.

AGRADECIMIENTOS

De Carlos

A mis padres, gracias por todo, por aguantarme todas mis malcriadeces, por apoyarme en los buenos y malos momentos, por estar siempre ahí para lo que necesitase, por confiar en mí, porque gracias a ellos hoy soy ingeniero.

A mi hermano que a pesar de ser pequeño siempre me ha ayudado y apoyado en todo, por ser el mejor hermano del mundo, gracias.

A mi familia que siempre han estado pendientes de mí, por todo el apoyo que me han dado pese a estar lejos, a mi tía baby que ha sido como una madre para mí, a mi tío Leonardo y a mi primo chicho quienes han sido mi padre y mi hermano aquí en La Habana.

A Marieta, quien realmente no existen palabras para agradecerle todo lo que ha hecho por mí, por darme los buenos consejos que siempre me dio, por hacer de mí una mejor persona, porque sin su apoyo nunca hubiese logrado este sueño, de veras muchas gracias por todo.

A mis amigos, quienes siempre me han estado apoyando en todo, a Julio por ser mi mejor amigo y aguantarme tanto en las buenas como en las malas, a Isis quien ha sido mi hermana que nunca tuve, por soportarme que sé ha sido un trabajo difícil, por toda su ayuda y su apoyo. A Mary por aguantarme durante todo este tiempo, a mi compañero de tesis el Robe, a todos los que he mencionado y los que no muchas gracias.

A mi tutora quien ha tenido que arrastrar conmigo durante todo este tiempo, por tener paciencia conmigo, por todo su apoyo para realizar este trabajo, muchas gracias profe.

A todos los profesores que de una forma u otra han contribuido al lograr este triunfo de hoy ser ingeniero, a Maigret, Hilda, Yeny, Yanet, a todas muchas gracias por su confianza.

A todas las personas que de una forma u otra han contribuido al cumplimiento de este sueño.

AGRADECIMIENTOS

De Roberto

A mis padres Julio y Milagros, por apoyarme en todas las metas que me he trazado en la vida y darme las fuerzas para seguir adelante, por creer en mi por todo su esfuerzo y sacrificio.

A mis hermanos Rosy y Braulio, por compartir y vivir conmigo cada momento de mi carrera y por darme la ardua tarea de ser su ejemplo a seguir.

A mis abuelos Cuca, Julio y Roberto, por estar siempre pendientes de mí y por hacerme tan feliz con su presencia.

A mis abuelos Orquidea y Jesús, por ser la luz que me inspiró desde mis primeros años de vida, por cuidarme e inculcarme tantos valores.

A mis primos Niurka y Yunerlin, por brindarme su apoyo y estar al tanto de cada detalle de mi trayectoria como estudiante, por compartir mis momentos de felicidad desde la distancia.

A mi linda novia, por su paciencia y apoyo incondicional, por su compañía, dedicación, cariño y amor.

A mi suegra, por quererme tanto y cocinarme tan rico, por demostrarme que se puede llegar a querer a una suegra.

A mis padrinos, por hacer todo lo posible para que llegara hasta donde estoy hoy, por poder contar con ellos cuando los necesitaba.

A mis tías, por ser partícipes de mis logros y por

A mis tutoras Mailen, Marieta y Reina, por el apoyo y la confianza depositada para la realización de este trabajo.

A mis profesores, por haber contribuido con sus conocimientos a mi formación.

A todos los amigos que me han acompañado durante estos largos 5 años de carrera.

RESUMEN

La atención a la población está dirigida, fundamentalmente, al control, tramitación, orientación y atención de quienes acuden o se dirigen a los organismos para plantear sus insatisfacciones. Actualmente, en la Facultad 3 de la Universidad de las Ciencias Informáticas (UCI) este proceso se realiza de forma manual, por lo cual existen dificultades que inciden negativamente en el adecuado control y seguimiento del mismo. Para darle solución a este problema, la presente investigación tiene como objetivo desarrollar un sistema informático que permita gestionar la información generada en este proceso, definiéndose la metodología, herramientas y tecnologías a usar para su implementación. Se muestran los resultados obtenidos en cada fase de la metodología, validándose en cada momento los artefactos generados. Como resultado se obtiene una aplicación informática que permite la gestión de las quejas, así como el control y seguimiento de toda la información que se genera.

Palabras claves: atención a la población, control y seguimiento, gestión de información, quejas.

Summary

The attention to the population is directed mainly to the control, transaction, guidance and attention of those who come or are directed to the organizations to present their dissatisfactions. Currently, at School 3 from the University of Informatics Sciences (UCI, by its acronym in Spanish) this process is done manually, whereby there are problems that adversely affect the proper control and monitoring of this process. To provide a solution to this problem, this research aims to develop a computer system that allows to manage the information generated in this process, defining the methodology, tools and technologies to use for its implementation. The results obtained at each stage of the methodology are displayed, validating at each time the generated artifacts. As a result, a software application that allows the complaints management, as well as the control and monitoring of all the information generated is obtained.

Key words: attention to the population, control and monitoring, information management, complaints

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La gestión de la información es un factor clave dentro de una empresa u organización. Su almacenamiento, tratamiento y uso constituyen algunos de los principales pilares para la toma de decisiones. Dentro de la información recogida en la empresa u organización se encuentran las quejas, estas son inconformidades, sugerencias u otros planteamientos realizados por los trabajadores pertenecientes a la misma.

Palabras cómo atención al cliente, consumidor, población, ciudadano son algunos de los términos más comunes usados para el manejo de la información que se genera a partir de las quejas recibidas. Usualmente se constituye una unidad dentro de la organización o se designa una persona con el objetivo de tratar la información recogida.

Actualmente este proceso se realiza mediante el uso de teléfonos, correo electrónico, correo postal o de forma personal. El avance de las tecnologías ha permitido la mejora en la prestación de este servicio.

En Cuba no existe una herramienta informática capaz de brindar estos servicios, ni hacer un adecuado procesamiento de la información obtenida.

El artículo 63 de la Constitución de la República de Cuba plantea que “Todo ciudadano tiene derecho a dirigir quejas y peticiones a las autoridades y a recibir la atención o respuestas pertinentes y en plazo adecuado, conforme a la ley.” (1) Este proceso es conocido como “Atención a la población”.

La atención a la población se considera un sistema donde se integran vías, procedimientos y acciones de interacción sistemática entre cuadros, funcionarios, docentes y no docentes del sector educacional. Está dirigida, fundamentalmente, al control, tramitación, orientación y atención de quienes acuden o se dirigen al organismo para plantear sus insatisfacciones. (2)

En la Facultad 3 de la Universidad de las Ciencias Informáticas el proceso de Atención a la Población presenta los siguientes problemas:

- La gestión de los datos de los compañeros que acuden a dirigir quejas o peticiones se realiza de forma manual, lo que algunas veces conlleva a que parte de la información se pierda y no se tenga un control exacto de la cantidad de personas que se han atendido así como de los temas tratados.
- La gestión de las quejas, inquietudes y peticiones se realiza de forma manual, lo que además de conllevar a pérdida de información no permite mantener un seguimiento adecuado de las respuestas que se le deben dar a los planteamientos. Estos mismos problemas se ven evidenciados en la realización de las notificaciones a todos los involucrados en el proceso (personal que solicita la atención, directivos que la brindan e implicados en las respuestas a cada inquietud o petición).
- La información que genera el proceso en cada nivel (decanato, vicedecanato, profesores principales, jefes de dpto., directores de centro), no se encuentra centralizada, lo que dificulta la creación de reportes que permitan dar una vista global o parcial del proceso en toda la

facultad, afectando además el control y la toma de decisiones.

Ante la problemática descrita anteriormente se define como **problema a resolver**: La forma en que actualmente se gestiona la información en el proceso de Atención a la población de la Facultad 3 incide negativamente en el control y seguimiento del proceso.

Teniendo como **objeto de estudio**: El proceso de Atención a la población de la educación superior.

Quedando definido como **objetivo general**: Desarrollar un sistema informático que permita gestionar la información generada en el proceso de Atención a la población de la Facultad 3 mejorando el control y seguimiento del proceso y como **objetivos específicos**:

- Realizar el marco teórico de la investigación sentando las bases para el desarrollo del sistema propuesto.
- Desarrollar el sistema informático propuesto de acuerdo a la metodología y tecnologías seleccionadas.
- Validar el correcto funcionamiento de la aplicación, así como las variables propuestas en la investigación.

Teniendo como **campo de acción**: Gestión de información para el proceso de Atención a la población en la educación superior.

Para dar cumplimiento a los objetivos propuestos se definen las siguientes **tareas a realizar:**

- Análisis de los principales conceptos asociados al proceso de Atención a la población.
- Análisis de los sistemas informáticos de gestión de información asociados a los procesos de Atención a la población y gestión de inquietudes existentes en Cuba.
- Análisis de las metodologías para el desarrollo de software identificando las adecuadas para realizar el sistema propuesto.
- Modelación del negocio teniendo en cuenta los artefactos propuestos por la metodología seleccionada para esta fase.
- Validación del negocio.
- Diseño de las funcionalidades del sistema informático teniendo en cuenta los artefactos propuestos por la metodología seleccionada para esta fase.
- Validación del diseño propuesto.
- Implementación de las funcionalidades del sistema informático haciendo uso de las tecnologías seleccionadas.
- Verificación del sistema informático mediante pruebas de caja blanca.
- Validación del sistema.
- Validación de las variables de la investigación.

Planteando como **idea a defender** que**:** El desarrollo de un sistema informático contribuirá a mejorar el control y seguimiento del proceso de atención a la población en la Facultad 3.

Para dar cumplimiento a las tareas propuestas se utilizan como **Métodos Científicos de la Investigación:**

Métodos Teóricos:

Permiten explicar los hechos y profundizar en las relaciones esenciales y cualidades fundamentales de los procesos, hechos y fenómenos. En ellos están comprendidos toda una serie de procedimientos que posibilitan la asimilación teórica de la realidad y que se adecuan a las condiciones en que se va a desarrollar la investigación. (3)

- **Analítico – Sintético:** Se utiliza para realizar un estudio de las soluciones similares existentes y sus usos en el ámbito nacional e internacional, así como las ventajas y desventajas que poseen.

Métodos Empíricos:

Los métodos de investigación empíricos conllevan toda una serie de operaciones prácticas con el objeto y los medios de investigación que permiten revelar las características y propiedades fundamentales y relaciones esenciales del objeto. (3)

- **La Entrevista:** Facilitó la obtención de información acerca de cómo se realiza el proceso en la Facultad, realizando encuentros con el cliente para la formulación de posibles soluciones.

El documento está estructurado de la siguiente manera:

Capítulo 1: Fundamentos Teóricos

En este capítulo se presentan los elementos teóricos que sirven de base a la investigación del problema planteado. Se hace un análisis de las soluciones similares existente al sistema que se desea implementar, los lenguajes de programación, herramientas, metodologías de desarrollo de software y tecnologías a usar en el desarrollo del sistema.

Capítulo 2: Propuesta de Solución

En este capítulo se presenta la solución propuesta con todos los aspectos definidos en la fundamentación teórica. Esta solución contiene los artefactos necesarios generados según la metodología escogida para desarrollar el sistema.

Capítulo 3: Validación de resultados

En este capítulo se realizan las validaciones de los resultados que se obtuvieron con el sistema ya en ejecución, haciendo uso de los métodos definidos en el marco teórico de la investigación. Además, se valida que el diseño realizado cumpla con la calidad requerida y que el sistema implementado satisface las necesidades de los clientes**.**

Capítulo 1. Fundamentación Teórica

En este capítulo se describen los elementos teóricos necesarios que sustentan el proceso de desarrollo de investigación del problema planteado. Se realiza un análisis de las metodologías de software, los lenguajes de programación, los elementos de la arquitectura, las metodologías y las herramientas a utilizar en el desarrollo del sistema con el propósito de justificar su uso, así como un estudio del arte.

1.1 Marco Teórico. Conceptos y Definiciones

Con el objetivo de lograr un mejor entendimiento del presente trabajo a continuación se enuncian los principales conceptos tratados. Estos conceptos se enmarcan en al ámbito de la Educación Superior.

Atención a la Población:

Se considera un sistema donde se integran vías, procedimientos y acciones de interacción sistemática entre cuadros, funcionarios, docentes y no docentes del sector educacional, dirigidas, fundamentalmente, al control, tramitación, orientación y atención de quienes acuden o se dirigen al organismo para plantear sus insatisfacciones. (2)

Queja:

Es la inconformidad sobre actuaciones de dirigentes, funcionarios por el personal docente o no docente en general, motivada por violaciones o irregularidades en el funcionamiento de la escuela, el uso inadecuado de métodos, procedimientos o decisiones tomadas con las que no está de acuerdo. (4)

1.2 Soluciones Existentes

Se realizó una investigación tanto a nivel nacional como internacional con el objetivo de determinar la existencia de sistemas informáticos que informaticen el proceso de atención a la población.

Nacionales

Oficina de atención al público de la Fiscalía General de la Republica (5)

La Fiscalía General cuenta con un departamento de Atención a la Población, que se encarga de investigar, tramitar y dar respuesta que en el orden legal proceda, a las denuncias, quejas y reclamaciones de la población.

Además existe una oficina que se encarga de la atención directa con el público, donde se recepcionan cada uno de los escritos que entreguen las personas y además existe un fiscal que las atiende en caso de que lo soliciten. Esta oficina pertenece al departamento de atención a la población, en la misma trabaja una recepcionista, que registra los datos de las personas que allí acuden, un asistente fiscal y un fiscal. Estos dos últimos se encargan de atender a las personas que deseen presentar su planteamiento directamente con el fiscal. Primeramente deben buscar los antecedentes de la persona para estar documentados y poder ofrecer un tratamiento más efectivo al promovente. El fiscal después de escucharlo siempre responde u orienta a la persona, mientras que el asistente fiscal toma nota de todo lo planteado, y si es necesario se realiza el acta de la entrevista.

Para el caso de aquellas personas que solo vienen a dejar escritos y no desean ver al fiscal, la recepcionista va acumulando dichos escritos y posteriormente los entrega en la oficina de correspondencia. La oficina de correspondencia se encarga de distribuir a las diferentes direcciones toda la documentación que llega.

Formulario para el planteamiento de quejas y solicitudes del Tribunal Supremo Popular (6).

El sistema de Tribunales Populares, en correspondencia con el principio de que la oportuna y adecuada atención a la población es un factor de vital importancia, tiene creado un mecanismo en función de atender cada inquietud, duda o inconformidad manifestada por el pueblo, y brindarle una respuesta adecuada. Dicha atención en los tribunales se ofrece todos los días hábiles en horario laborable (6).

En el Tribunal Supremo Popular existe una Oficina de Atención a la Población, donde radican supervisores de la actividad judicial con experiencia en el trabajo de los tribunales, encargados de orientar y tramitar los planteamientos presentados. En los tribunales provinciales y municipales, las personas son atendidas por el presidente, el secretario judicial u otro funcionario designado. El término para la tramitación de los asuntos es de hasta 60 días hábiles (6).

Las estructuras organizativas del Sistema de Tribunales colaboran entre sí, en pos de la prontitud y calidad requeridas. Periódicamente, se evalúa el comportamiento de esta actividad, a partir de las causas que generan inconformidades, tendencias fundamentales que se manifiestan y niveles de responsabilidad (6).

Formulario de quejas y reclamaciones del Ministerio de las Comunicaciones. (7)

El Ministerio de Comunicaciones cuenta con la posibilidad de enviar las no conformidades de los ciudadanos con algún proceso en el cual haya estado envuelto y cuyo nivel de satisfacción no sea el deseado. En ese espacio se le orienta a los interesados en usar el servicio cómo contactar a las diferentes instituciones con las que cuenta el ministerio, a qué lugares debe la persona dirigirse para expresar inquietudes, preguntas, dudas, insatisfacciones, sugerencias, denuncias, comentarios y otras incidencias que desee compartir.

A través de “Atención Ciudadana” usted podrá ejercer su derecho constitucional al expresar las irregularidades, quejas y reclamaciones asociadas al sistema nacional de comunicaciones. Es un espacio que facilita la retroalimentación con el usuario, la respuesta ágil y eficiente a sus inconformidades, así como la mejora continua de los productos y servicios (7).

Internacionales

Formulario de Reclamaciones del Banco de la Nación en Perú. (8)

El Banco de la Nación brinda servicios a las entidades estatales, promueve la bancarización y la inclusión financiera en beneficio de la ciudadanía complementando al sector privado, y fomenta el crecimiento descentralizado del país, a través de una gestión eficiente y auto-sostenible. Su objetivo es ser reconocido como socio estratégico del Estado Peruano para la prestación de servicios

financieros innovadores y de calidad, dentro de un marco de gestión basado en prácticas de Buen Gobierno Corporativo y gestión del talento humano (8).

Dentro de las principales expectativas del Banco se encuentra la satisfacción del cliente con los servicios brindados por la institución, es por eso que se ha encargado de habilitar varias vías a través de las cuales los ciudadanos pueden expresar sus inconformidades con algún servicio de los brindados por la entidad. Entre las vías que cuentan se encuentra habilitada una línea gratuita las 24 horas del día de los 365 días del año. Cuentan además con una red de agencias encargadas de realizar el proceso de atención a la población con todas aquellas personas que acudan a solicitarlo personalmente y en su portal web brindan un formulario de reclamaciones para aquellas personas que deseen enviar sus quejas de forma online.

Valoración de los sistemas estudiados

Los sistemas analizados anteriormente además de estar elaborados para atender el proceso de atención a la población en un ámbito muy específico no satisfacen las necesidades existentes en la Facultad 3, ya que no permiten la asignación de las quejas por las diferentes áreas administrativas por las que se estructura, ni gestionar el flujo de estados por los que van pasando desde el momento en que se originan hasta el que se da la correspondiente respuesta a la persona que la generó. Estas soluciones no posibilitan lograr una interacción entre los implicados en el proceso de atención a la población y carecen además de la posibilidad de generar diferentes reportes que permiten conocer si es correcta o no la gestión de las quejas y reclamaciones, así como el nivel de conformidad con las respuestas brindadas.

1.3 Metodologías, Herramientas y Tecnologías Utilizadas

A continuación se presentan las descripciones de las herramientas, metodologías y tecnologías utilizadas para dar solución al problema. Para definir la selección se tomaron en cuenta las necesidades existentes, tendencias actuales y el entorno donde se desplegará el sistema.

1.3.1 Metodología de Desarrollo

Una metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. (9)

Desarrollar un buen software depende de un gran número de actividades y etapas, donde el impacto de elegir la metodología para un equipo en un determinado proyecto es trascendental para el éxito del producto.

Según la filosofía de desarrollo se pueden agrupar las metodologías en dos enfoques. Las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado.

Metodologías ágiles:

- FDD: DESARROLLO MANEJADO POR RASGOS
- XP: EXTREME PROGRAMMING
- SCRUM
- AUP

Metodologías tradicionales:

- MFS: MICROSOFT SOLUTION FRAMEWORK
- RUP: RATIONAL UNIFIED PROCESS

Para el desarrollo del sistema se hace necesaria la selección de una metodología adecuada a las características del proyecto que sea capaz de brindar ventajas al equipo de desarrollo.

Para la selección del enfoque a usar para el desarrollo del sistema se realizó una comparación de las principales características ofrecidas por cada uno de ellos (Ver Tabla 1) y las características que presenta el proyecto, decidiéndose hacer uso de un enfoque ágil para el desarrollo ya que es el que más se adecúa a las características del proyecto.

Tabla 1: Tabla comparativa entre el enfoque tradicional y ágil.

Metodologías Ágiles	Metodologías Robustas
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.

Principales características del proyecto:

- Equipo de desarrollo pequeño (solo 2 personas).
- Proyecto pequeño y de corta duración (solo 6 meses).
- Existe un constante intercambio con el cliente

Los autores del trabajo seleccionaron dentro de este enfoque como metodología de desarrollo EXTREME PROGRAMMING (XP por sus siglas en inglés).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP cuenta con un conjunto de valores que son importantes para el logro del producto, entre los que se encuentran:

Valores de XP (10)

- Comunicación: Crear software requiere de sistemas comunicados.

- Simplicidad: Empezar con lo necesario y requerido y trabajar desde ahí.
- Retroalimentación: Del sistema, del cliente, y del equipo.
- Respeto: El equipo debe trabajar como uno, sin hacer decisiones repentinas.

Cuenta también con las siguientes características y artefactos, los cuales son necesarios en el desarrollo de la metodología.

Características fundamentales (10)

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Frecuente 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, es decir, rescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Artefactos esenciales de XP propuestos a generar (10)

- Historias del usuario.
- Tareas de ingeniería.
- Pruebas de aceptación.
- Pruebas unitarias y de integración.
- Plan de iteraciones
- Código.
- Casos de prueba.

Descripción de los artefactos propuestos a generar

- **Historias del usuario:** Las historias de usuario (HU) son la técnica utilizadas en XP para la descripción de los requisitos del software. Representan una breve descripción del comportamiento del sistema. Emplea terminología del cliente sin lenguaje técnico para describir brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Se emplean para hacer estimaciones de tiempo y son la base para las pruebas de unidad y presiden la creación de las pruebas de aceptación.

Las historias de usuario deben ser: (11)

- Independientes unas de otras: De ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes.
- Negociables: La historia en sí misma no es lo suficientemente explícita como para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.
- Valoradas por los clientes o usuarios: Los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos más que para el desarrollador.
- Estimables: Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla. Esto permite estimar el tiempo total del proyecto.
- Pequeñas: Las historias muy largas son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.
- Verificables: Las historias de usuario cubren requisitos funcionales, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

- **Tareas de ingeniería:** Las tareas de ingeniería son un conjunto de acciones a desarrollar para resolver las historias de usuario. Permiten organizar el proceso de implementación además de posibilitar que sea conocido el grado de complejidad de cada historia de usuario teniendo en cuenta la cantidad de tareas asociadas a ella.
- **Pruebas de aceptación:** constituyen pruebas basadas en la ejecución, revisión y retroalimentación de las funcionalidades que han sido diseñadas para el software. Estas pruebas se realizan a través de modelos de prueba conocidos como casos de prueba.
- **Pruebas unitarias:** constituyen una forma de probar el correcto funcionamiento de un módulo de códigos, por lo que también se le conoce como pruebas modulares. Tienen como objetivo asegurar que cada uno de los módulos de códigos funcione correctamente por separado.
- **Plan de iteraciones:** define exactamente cuáles historias de usuario serán implementadas en cada iteración. Se toma como base cada una de las historias de usuarios y el esfuerzo que se requiere para el desarrollo de estas, y se procede a fragmentar el trabajo en iteraciones.

1.3.2 Lenguaje de Modelado (UML 2.0)

El lenguaje unificado de modelado (UML por sus siglas en inglés) es el estándar más utilizado para especificar y documentar cualquier sistema de forma precisa. Es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema. UML proporciona una forma

estándar de escribir los planos de un sistema, cubriendo tanto las cosas conceptuales (procesos del negocio y funciones del sistema), como las cosas concretas (las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes de software reutilizables). (12)

UML utiliza una serie de diagramas para la modelación del software los cuales se agrupan en tres grandes grupos: (13)
Estructura

- Diagrama de clases
- Diagrama de objetos
- Diagrama de componentes
- Diagrama de estructura compuesta
- Diagrama de paquetes
- Diagrama de despliegue

Comportamiento

- Diagrama de casos de uso
- Diagrama de actividades
- Diagrama de estado

Interacción

- Diagrama de secuencia
- Diagrama de colaboración UML 1.X / Diagrama de comunicación UML 2.0
- Diagrama de tiempo
- Diagrama de interacción

Estos diagramas soportan el ciclo de desarrollo de todo el software haciéndose efectivo su uso durante todas las etapas del desarrollo.

1.3.3 Herramienta CASE

Ingeniería de Software Asistida por Computación (CASE por sus siglas en inglés) son sistemas de software que proporcionan ayuda automatizada a las actividades del proceso del software, como el análisis de requisitos, el modelado de sistemas, la depuración y las pruebas. (14)

Visual Paradigm 8.0

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar

documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (15)

Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Dentro de sus principales características se encuentran (16)**:**

- Generador de documentación.
- Disponible en múltiples plataformas (Windows, Linux).
- Generador de código de base de datos.
- Provee el modelado de procesos de negocios.
- Proporciona el código y compatibilidad hasta con 10 lenguajes.
- Descripción detallada de sus características.

Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del mismo. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software**.**

1.3.4 Entorno de Desarrollo Integrado

Un IDE[1] es un programa informático compuesto por un conjunto de herramientas (un editor de código, un compilador, un depurador y un constructor de interfaz gráfica) para programar en un lenguaje de programación o en varios, que van a facilitar las tareas de desarrollo y mantenimiento de las aplicaciones. (17)

NetBeans 8.0

NetBeans IDE es un entorno de desarrollo, una herramienta mediante la cual los programadores pueden escribir, compilar, depurar y ejecutar programas.

Es un proyecto de código abierto, libre y gratuito sin restricciones de uso tanto comercial como no comercial. Está escrito en java pero puede servir para cualquier otro lenguaje de programación. Existe además un gran número de módulos para extenderlo.

Cuenta con una comunidad en constante crecimiento lo que ha contribuido al aumento paulatino de sus prestaciones así como a la disminución de errores de programación que pudiesen existir. Es fácil de instalar, de uso instantáneo y es soportado por varios sistemas operativos. (18)

1.3.5 Marcos de Trabajo

Un marco de trabajo (framework en inglés) para el desarrollo de aplicaciones web, los cuales facilitan el trabajo de la programación, disminuyen el tiempo de implementación y optimizan el código.

[1] IDE: Entorno de desarrollo integrado por sus siglas en inglés

Un marco de trabajo, simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además proporcionan estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilitan la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (19)

Symfony2 2.3.7

Symfony2 ha sido ideado para utilizar todas las características de PHP 5.3 y por eso es uno de los frameworks PHP[2] con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto (20). Es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web, además proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. (19)

JQuery 2.1

Es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM[3], manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX[4] a páginas web. (21)

Ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. Se basa fundamentalmente en la idea de escribir menos y hacer más, tributando a un código más organizado, legible y de fácil mantenimiento.

Bootstrap 3.2

Es un marco de trabajo desarrollado para simplificar la creación de diseños web. Combina CCS y JavaScript para lograr una interfaz agradable y vistosa. La mayor ventaja es que se puede crear interfaces que se adapten a los distintos navegadores con el apoyo de un marco de trabajo con numerosos componentes web que ahorrarán mucho esfuerzo y tiempo. (22)

Entre las características principales de Bootstrap se encuentran:

- Ofrece una serie de plantillas CSS y ficheros JavaScript que facilitan la integración del marco de trabajo de forma sencilla en los proyectos webs.
- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tabletas y móviles a distintas escalas y resoluciones.
- Se integra perfectamente con las principales bibliotecas de JavaScript, por ejemplo JQuery.
- Ofrece un diseño robusto usando estándares como CSS3/HTML5.
- Es un marco de trabajo ligero que se integra de forma sencilla con el proyecto actual.

[2] PHP: Preprocesador de Hipertexto
[3] DOM: Document Object Model en español Modelo en Objetos para representación de Documento
[4] AJAX: Asynchronous JavaScript + XML en español JavaScript asíncrono + XML

- Funciona con todos los navegadores, incluido Internet Explorer usando HTML para que reconozca las etiquetas de HTML5.
- Dispone de distintas capas redefinidos con estructuras fijas a 940 píxeles de distintas columnas o diseños fluidos.

1.3.6 Lenguajes de Programación

HTML 5

HyperText Markup Language (HTML por sus siglas en inglés) en español lenguaje de marcado de hipertexto, en su versión 5, es usado para describir la estructura y el contenido en forma de texto. Este puede describir hasta un cierto punto, la apariencia de un documento. Además se ha convertido en el formato más fácil para la creación de páginas web debido a su sencillez y a que no hay que compilar el código para ver si funciona. Se puede ver en forma inmediata el resultado del trabajo y también es usado para complementar el texto con objetos tales como imágenes. (23)

Javascript

Es un lenguaje interpretado e independiente de plataforma que permite incluir macros en páginas web. Estas macros se ejecutan del lado del cliente y no en el servidor. La característica de JavaScript que más simplifica la programación es que, aunque el lenguaje soporta varios tipos de datos, no es necesario declarar el tipo de las variables, argumentos de funciones ni valores de retorno de las funciones. El tipo de las variables cambia implícitamente cuando es necesario, lo que ayuda a programar con rapidez macros sencillas. (24)

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

CSS 3

Cascade Style Sheets (CSS) por sus siglas en inglés, es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML[5] y XHTML[6]. CSS es la mejor forma de separar los contenidos de su presentación, y en la actualidad está muy ligado a la creación de páginas web complejas (25)

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados *"documentos semánticos"*). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (25)

PHP 5.6

Es un acrónimo recursivo que significa “PHP Hypertext Pre-processor”, (inicialmente se llamó Personal Home Page). PHP es un lenguaje de script interpretado en el lado del servidor utilizado

[5] HTML: Hyper Text Markup Language, en español Lenguaje de Marcado de Hipertexto.
[6] XHTML: Versión avanzada de HTML y basada en XML

para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. No necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado un servidor web con las librerías de PHP. (26)

Twig

Es un motor de plantillas para el lenguaje PHP que se caracteriza por ser (27):

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

1.3.7 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos en inglés DBMS: Data Base Management System (SGBD por sus siglas en inglés) es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarias para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. (28)

PostgreSQL 9.3

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[7] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más utilizado del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (29)

Dentro de las principales características que identifican a PosgreSQL están (29):

- Es una base de datos 100% ACID (Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad).
- Integridad referencial
- Copias de seguridad en caliente (Online/hot backups)
- Unicode
- Juegos de caracteres internacionales
- Regionalización por columna
- Múltiples métodos de autentificación
- Acceso encriptado vía SSL
- Completa documentación

[7] BSD: Berkeley Software Distribution, en español distribución de software berkeley es una licencia de software libre permisiva.

- Multiplataforma.

1.3.8 ORM Doctrine 2.0

Es un asignador objeto relacional (ORM) para PHP 5.3.0+ que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una capa de abstracción de base de datos (DBAL por Data Base Abstraction Layer). La principal tarea de los asignadores objetos relacionales es la traducción transparente entre objetos (PHP) y las filas relacionales de la base de datos. Una de las características clave de Doctrine es la opción de escribir las consultas de base de datos en un dialecto SQL propio orientado a objetos llamado Lenguaje de Consulta Doctrine (DQL por Doctrine Query Language). (30)

Otra característica de Doctrine es el bajo nivel de configuración que necesita para empezar un proyecto. Doctrine puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidades extras a las clases autogeneradas.

1.3.9 Servidor Web

La piedra angular de cualquier sitio o portal Web es, con toda seguridad, su servidor Web; el software encargado de atender las peticiones de los clientes y ejecutar las páginas Web solicitadas. Existen multitud de aplicaciones para montar servidores Web, muchos de ellos distribuidos como Software libre y siendo, sin lugar a dudas, el más popular de todos: Apache. (31)

Apache 2.2

El servidor HTTP[8] Apache es un servidor web HTTP el cual se encuentra bajo la licencia de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1. Es altamente configurable y extensible con módulos de terceros, proporciona el código fuente completo y viene con una licencia sin restricciones. Es una aplicación que está en constante desarrollo y alienta los comentarios de los usuarios a través de nuevas ideas, informes de errores y parches. (32)

1.3.10 Control de Versiones

Un sistema de control de versiones *(o* sistema de control de revisiones*)* es una combinación de tecnologías y prácticas para seguir y controlar los cambios realizados en los ficheros del proyecto, en particular en el código fuente, en la documentación y en las páginas web. (33)

Subversion 1.8.8

Subversion es un sistema centralizado de control de versiones que permite realizar el seguimiento de los cambios en archivos empleados en proyectos de software. Basado en el trabajo inicial de CVS (Concurrent Versions System), constituye una implementación más eficiente de este último y ha conseguido mayor popularidad en proyectos de software libre y abierto y en organizaciones empresariales. (34)

[8] HTTP: Protocolo de Transferencia de Hipertexto

1.4 Arquitectura de Software

Se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para la guía en el desarrollo de software.

Es la organización fundamental de un sistema encarnado en sus componentes, las relaciones de los componentes con cada uno de los otros y con el entorno, y los principios que orientan su diseño y evolución (35).

1.4.1 Patrones Arquitectónicos

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos, ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor. (36)

Modelo-Vista-Controlador

El Modelo–Vista–Controlador (MVC por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el **modelo**, la **vista** y el **controlador**, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. (37)

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

De manera genérica, los componentes de MVC se definen como sigue (38):

- **El Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.
- **El Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información. También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta de

'modelo', por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo' .

- **La Vista:** Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

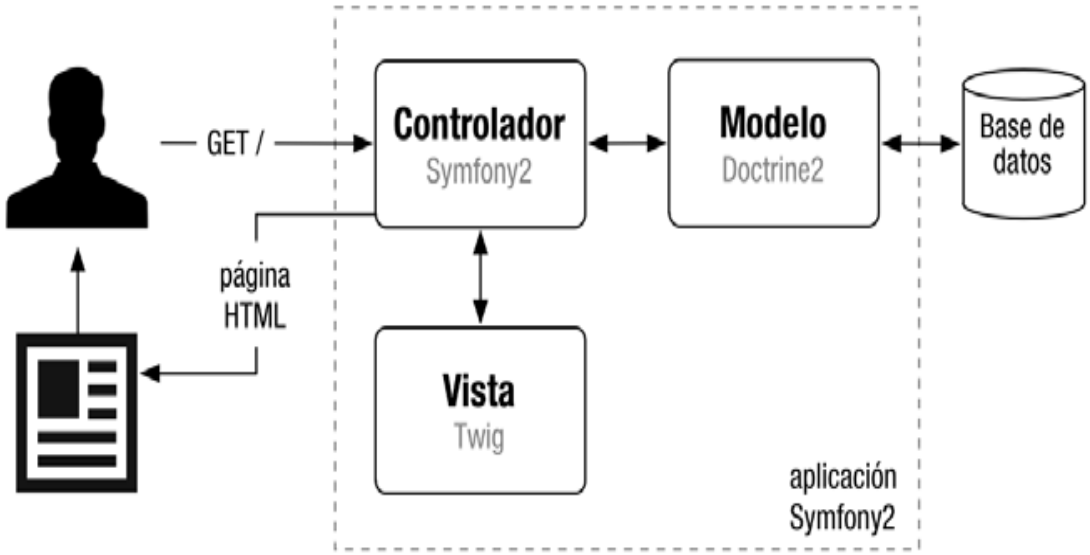


Ilustración 1: Representación del patrón MVC en Symfony2 (20)

1.5 Patrones de Diseño

El patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. (39)

Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software. Todo lo contrario: intentan codificar el conocimiento, las expresiones y los principios ya existentes. (39)

Existen dos clasificaciones en las que se agrupan, los patrones GRASP (General Responsibility Asignment Software Patterns) en español Patrones Generales de Software para Asignar Responsabilidades, los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones y los patrones GOF (Gang Of Four) que solucionan problemas de creación de instancias.

1.6 Ingeniería de Requisitos

La ingeniería de requisitos proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar

la solución sin ambigüedades, validar la especificación, y administrar los requisitos conforme estos se transforman en un sistema operacional. (40)

Los requisitos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento. (14)

Técnicas de Captura

Para la captura de los requisitos tantos funcionales como no funcionales las técnicas que se aplicarán serán la entrevista (abierta y cerrada) y la tormenta de ideas (14)**.**

1.7 Técnicas de Validación

Para garantizar la calidad del análisis y el diseño se proponen un conjunto de métricas que permitirán medir la calidad de la especificación de los requisitos y del diseño propuesto en la solución, así como evaluar la calidad de los atributos internos del sistema.

1.7.1 Validación de Requisitos

La validación de requisitos trata de mostrar que estos son realmente definen el sistema que el cliente desea. (14)

Con el objetivo de verificar si los requisitos del software obtenidos definen el sistema que el cliente desea, se llevará a cabo un proceso de validación, para el cual se emplearán las siguientes técnicas.

- **Revisiones de requisitos**: En este punto los requisitos son analizados sistemáticamente por un equipo de revisores. (14)
- **Construcción de prototipos:** En este enfoque de validación, se muestra un modelo ejecutable del sistema a los usuarios finales y a los clientes, así estos pueden experimentar con este modelo para ver si cumple con sus necesidades reales. (14)

1.7.2 Validación del Diseño

Para la validación del diseño se usarán las métricas basadas en clases: Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC). El resultado de aplicar ambas métricas permitirá la validación del diseño propuesto en la investigación.

Tamaño Operacional de Clases (TOC): consiste en un cálculo de los atributos u operaciones totales que se realizan en las clases, esta métrica mide el grado de responsabilidad, complejidad y reutilización que poseen las clases.

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

Relaciones entre Clases (RC): está dada por el número de relaciones de uso de una clase con otras. La aplicación de dicha métrica permite evaluar atributos como el Acoplamiento, la Complejidad de mantenimiento, la Reutilización y la Cantidad de pruebas.

- **Acoplamiento:** Un aumento del RC implica un aumento del Acoplamiento de la clase.

- **Complejidad de mantenimiento:** Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas:** Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

1.8 Pruebas de Calidad

Las pruebas son un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática con el objetivo de detectar errores en el software. (40)

Hay dos maneras de probar cualquier producto construido: uno si se conoce la función específica para la que se diseñó el producto, se aplican pruebas que demuestren que cada función es plenamente operacional, mientras que se buscan los errores de cada función; dos si se conoce el funcionamiento interno, se aplican pruebas para asegurarse que "todas las piezas encajan", es decir que las operaciones internas se realizan de acuerdo con las especificaciones, y se han probado todos los componentes internos de manera adecuada. Al primer enfoque de prueba se le denomina prueba de caja negra, y al segundo prueba de caja blanca. (40)

XP propone como pruebas a realizar las pruebas unitarias y las pruebas de aceptación las cuales serán llevadas cabos bajo casos de prueba previamente diseñados como lo propone la metodología.

1.8.1 Pruebas Unitarias (caja blanca)

La prueba de caja blanca del software se basa en un examen cercano al detalle procedimental. Se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de pruebas que ejerciten conjuntos específicos de condiciones, bucles o ambos. Al emplear los métodos de prueba de caja blanca el ingeniero del software podrá derivar casos de prueba que: 1) garanticen que todas las rutas independientes dentro del módulo se han ejercido al menos una vez. 2) ejerciten los lados verdadero y falso de todas las decisiones lógicas. 3) ejecuten todos los bucles en sus límites y dentro de sus límites operacionales y 4) ejerciten estructuras de datos internos para asegurar su validez. (40)

Técnica del camino básico

Para comprobar que cada sentencia de código se ejecuta al menos una vez, se realizaron pruebas al código de las funcionalidades más complejas desde el punto de vista de la programación.

La prueba del camino básico es una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe. El método del camino básico permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. (40)

Para obtener los casos de prueba a partir la técnica seleccionada se debe construir el grafo de flujo correspondiente al código de la funcionalidad. Luego se determina la complejidad ciclomática V (G) del grafo resultante, la cual es un indicador del número de caminos independientes que existen en

un grafo, es decir, es cualquier camino dentro del código que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición. (40)

La complejidad ciclomática V (G) de un grafo se define como:

V (G) = E – N + 2, donde E es el número de aristas, y N el número de nodos de la gráfica de flujo.

1.8.2 Pruebas Funcionales (caja negra)

Las pruebas de caja negra, también denominadas, pruebas de comportamiento, se concentran en los requisitos funcionales del software. Permiten derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales del programa. (40)

La prueba de caja negra intenta encontrar errores de las siguientes categorías (40):

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

Técnica de partición de equivalencia.

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. (40)

Los casos de pruebas que se diseñan para este tipo de técnica se basan en una evaluación de las clases de equivalencia para una condición de entrada.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. (40)

Las clases de equivalencia se definen de acuerdo a las siguientes directrices: (40)

- Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada requiere un valor especifico, se define una clase de equivalencia válida y una no válida.
- Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.
- Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida

Al aplicar estas directrices para la derivación de clases de equivalencia, se desarrollarán y se ejecutarán los casos de prueba para cada objeto de los datos del dominio de entrada.

CONCLUSIONES DEL CAPÍTULO

Luego de realizado el marco teórico de la investigación se arriba a las siguientes conclusiones:

- Los sistemas informáticos analizados incluyen de una forma u otra el proceso de Atención a la Población pero no se corresponden con las necesidades de la facultad por lo que se fundamenta la necesidad de desarrollar una nueva solución.
- El análisis de las metodologías, herramientas y tecnologías realizado permitió la selección de las más adecuadas para el desarrollo de la solución propuesta.

Capítulo 2.Propuesta de Solución

En el presente capítulo se hace una descripción de la propuesta de solución mediante las tres primeras fases que define la metodología XP (Planificación, Diseño e Implementación) y los principales artefactos a crear en cada una de estas. Entre los artefactos que se generarán se encuentran las historias de usuarios, el plan de iteraciones y las tareas de ingeniería, para obtener de esta forma un producto con la calidad requerida por el usuario final.

Descripción de los procesos del negocio

Para lograr una mejor comprensión del negocio a continuación se hace una breve descripción de las características y el funcionamiento de los principales procesos que intervienen en el negocio:

- **Gestión de quejas:** El proceso de gestión de quejas es mediante el cual se realiza la creación, modificación, actualización, eliminación y el listado de las quejas que son generadas en el sistema, este proceso involucra a todas las personas que intervienen en el negocio.
- **Asignación de tiempo de respuesta y área a la queja:** Este proceso se lleva a cabo una vez recopiladas las quejas en el sistema, consiste en la asignación de una o varias áreas y el tiempo en que estas deberán ser respondidas por la persona encargada de darle seguimiento en el área a la que fue asignada, hasta llegar al fin del proceso el cual culmina cuando se le da respuesta a la misma. Para la realización de este proceso primeramente las quejas deberán haber sido evaluadas por la decana quien una vez haya revisado cada una le asignara los permisos de modificación a su asesora, persona encargada de asignarle el área a la que será enviada para darle seguimiento así como el tiempo en el que se le deberá dar una respuesta.

2.1 Planificación

La planificación en XP está basada en un conjunto de decisiones tomadas por el cliente de conjunto con los programadores. En esta primera fase se debe hacer primero una recopilación de todos los requisitos del proyecto, también debe haber una interacción con el usuario, y se debe planificar bien entre los desarrolladores que es lo que se quiere para así lograr los objetivos finales.

2.1.1 Requisitos del Sistema

Luego de la aplicación de las técnicas explicadas en el Capítulo 1 para la obtención de los requisitos del sistema se obtuvieron los siguientes.

Requisitos Funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar ante entradas particulares y de cómo se debe comportar en situaciones particulares. (14)

Se identificaron 61 requisitos funcionales agrupados en 20 historias de usuarios. A continuación se muestran los requisitos obtenidos con la historia de usuario a la que pertenecen y su prioridad para el negocio. En el caso del requisito funcional #3 Asignar tiempo de tramitación la determinación de este tiempo fue dada por el cliente (máxima autoridad en la facultad) al carecer de un documento legal que regule el tiempo de tramitación de las inquietudes planteadas según el tema tratado.

Tabla 2: Agrupación de requisitos funcionales por historias de usuario.

Historia de Usuario	Requisitos Funcionales	Prioridad
HU_1: Autenticar usuario	RF#1_ Autenticar usuario	Alta
HU_2: Asignar área a la queja	RF#2_ Asignar área a la queja	Alta
HU_3: Asignar tiempo de tramitación a la queja	RF#3_ Asignar tiempo de tramitación.	Alta
HU_4: Gestionar queja	RF#4_ Listar quejas RF#5_ Crear queja RF#6_ Modificar queja RF#7_ Eliminar queja RF#8_ Actualizar queja	Alta
HU_5: Consultar quejas por categorías	RF#9_ Consultar quejas por estudiantes RF#10_ Consultar quejas por trabajadores	Baja
HU_6: Gestionar organigrama de la facultad	RF#11_ Modificar organigrama RF#12_ Visualizar organigrama	Alta
HU_7: Consultar quejas generadas por factor	RF#13_ Consultar quejas por brigada RF#14_ Consultar quejas por UJC-CB RF#15_ Consultar quejas por PCC-Núcleo RF#16_ Consultar quejas por Sección Sindical	Media
HU_8: Consultar quejas por estado	RF#17_ Consultar quejas nuevas RF#18_ Consultar quejas en trámite RF#19_ Consultar quejas respondidas	Media
HU_9: Consultar quejas asignadas por áreas	RF#20_ Listas quejas por Departamento RF#21_ Listar queja por Vicedecanato RF#22_Listar queja por Centro de Producción RF#23_ Listar queja por Secretaria docente RF#24_ Listar quejas por Profesores Principales	Media
HU_10: Mostrar perfil personal	RF#25_ Mostrar perfil	Alta

HU_11:Gestionar nomencladores	RF#26_ Crear nomenclador RF#27_ Modificar nomenclador RF#28_ Eliminar nomenclador RF#29_ Actualizar nomenclador RF#30_ Listar nomencladores	Alta
HU_12: Ver detalles de queja	RF#31_ Ver detalles de queja	Alta
HU_13: Generar alertas	RF#32_ Generar alerta de vencimiento de tiempo de tramitación RF#33_ Generar alerta de asignación de nuevas quejas RF#34_ Generar alerta de respuesta de queja	Media
HU_14: Consultar quejas generadas por áreas	RF#35_Listas quejas por Departamento RF#36_Listar queja por Vicedecanato RF#37_Listar queja por Centro de Producción RF#38_Listar queja por Secretaria docente	Media
HU_15: Reporte de estado de las quejas asignadas por áreas y factores	RF#39_ Realizar reporte de estado de las quejas asignadas por departamento RF#40_ Realizar reporte de estado de las quejas asignadas por vicedecanato RF#41_ Realizar reporte de estado de las quejas asignadas por centro de producción RF#42_ Realizar reporte de estado de las quejas asignadas por profesor principal RF#43_ Realizar reporte de estado de las quejas asignadas a la secretaria docente RF#44_ Realizar reporte de estado de las quejas asignadas a la UJC RF#45_ Realizar reporte de estado de las quejas asignadas al Núcleo del PCC RF#46_ Realizar reporte de estado de las quejas asignadas a la Sección Sindical	Media
	RF#47_ Listar quejas por UJC-CB	

HU_16: Consultar quejas asignadas por factor	RF#48_ Listar quejas por PCC-Núcleo RF#49_ Listar quejas por Sección Sindical	Media
HU_17: Gestionar estudiante por grupo	RF#50_ Añadir estudiante RF#51_ Eliminar estudiante RF#52_ Listar estudiantes	Alta
HU_18: Gestionar estudiante por Comité de Base	RF#53_ Añadir estudiante RF#54_ Eliminar estudiante RF#55_ Listar estudiantes	Alta
HU_19: Gestionar Trabajador por Comité de Base	RF#56_ Añadir trabajador RF#57_ Eliminar trabajador RF#58_ Listar trabajador	Alta
HU_20: Gestionar Trabajador por Factor	RF#59_ Añadir trabajador RF#60_ Eliminar trabajador RF#61_ Listar trabajador	Alta

Requisitos no Funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. (14)

Estos requisitos especifican o restringen las propiedades emergentes del sistema, pueden especificar el rendimiento, la protección, la disponibilidad y otras propiedades del sistema. (14)

El incumplimiento de un requisito no funcional puede significar que el sistema entero sea inutilizable ya que estos son más críticos en ocasiones que los requisitos funcionales debido a que los usuarios normalmente pueden encontrar formas de trabajar alrededor de una función del sistema que realmente no cumple sus necesidades pero no ante la falla de un requisito no funcional que sea crítico para el funcionamiento del sistema. (14)

Se identificaron 21 requisitos no funcionales lo cuales para un mejor entendimiento se agruparon por las siguientes clasificaciones:

Requisitos de Software

- Un servidor web con los módulos: php5-xsl, php5-pgsql, php5-mencache, php5-cli, php- apc, php-soap, el php5-intl y php5-ldap.
- Un servidor de base de datos PostgreSQL en su versión 9.1 o superior.
- Computadoras clientes con un navegador web (Firefox 20.0 o superior).

Requisitos de Hardware

- En el cliente se requiere una pc con 256 MB de RAM como mínimo, procesador Intel® a 1 GHz de velocidad de procesamiento o superior, tarjeta de red Ethernet.
- Todas las máquinas implicadas en la funcionalidad de la aplicación deben estar conectadas a la red.
- El servidor requiere mínimo 1024 MB de RAM, procesador a 3 GHz de velocidad y tarjeta de red Ethernet.

Requisitos de Seguridad

Para abordar estos requisitos de seguridad se tuvieron en cuenta los siguientes aspectos y principios relacionados con la seguridad informática para garantizar una mayor calidad en cuanto a la seguridad del software.

Mínimo privilegio: se deben otorgar los permisos estrictamente necesarios para efectuar las acciones que se requieran.

Participación universal: la gestión de la seguridad informática necesita de la participación de todo el personal de una institución.

Proporcionalidad: las medidas de seguridad deben estar en correspondencia con lo que se protege y con el nivel de riesgo existente.

Confidencialidad: la información o los activos informáticos son accedidos solo por las personas autorizadas para hacerlo.

Integridad: los activos o la información solo pueden ser modificados por las personas autorizadas y de la forma autorizada.

Disponibilidad: los activos informáticos son accedidos por las personas autorizadas en el momento requerido. (41)

- El sistema podrá ser utilizado solamente por usuarios autenticados en el mismo.
- La autenticación será la primera acción en el sistema, proporcionándole los privilegios concebidos al usuario, de forma tal que la información sensible sea vista y manejada por la persona adecuada.
- El administrador es el único que tiene el control total del sistema.
- Los cambios en el sistema sólo pueden ser realizados por los usuarios con los permisos requeridos para esta acción.
- El sistema será usado en principio solamente por el personal perteneciente a la Facultad 3.

Requisitos de Interfaz

- El sistema deberá ser independiente de la plataforma.
- Los colores serán estándares en toda la aplicación.

Requisitos de Usabilidad

La usabilidad se define como la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso. La usabilidad es una cualidad abstracta por lo cual no puede ser medida directamente, se descompone habitualmente en atributos Algunos de estos atributos de usabilidad son (42):

Facilidad de Aprendizaje: indica qué tan fácil es aprender la funcionalidad básica del sistema, como para ser capaz de realizar correctamente las tareas que desea llevar a cabo cualquier tipo de usuario.

Presentación visual apropiada: la interfaz gráfica es una parte importante del sistema, y un buen diseño de la misma puede hacer que un sistema aumente su nivel de usabilidad.

- El sistema deberá poseer una interfaz web sencilla, lo más atractiva y clara posible para el usuario, además de poder ser usado por cualquier persona con conocimientos básicos en el manejo de la computadora y el entorno Web en sentido general.
- El software tendrá en su portada inicial las políticas de uso concebidas para hacer uso de él por cualquier usuario.

Requisitos de Portabilidad

- El sistema estará desarrollado con tecnologías multiplataforma permitiendo que el servidor sea instalado tanto en Windows como en Linux, llevando a cabo esta operación sin necesidad de efectuar cambios significativos.

Requisitos de Disponibilidad

- El sistema podrá ser usado en cualquier momento por todos los usuarios autorizados.
- Para poder hacer uso del sistema deberá existir conexión.

2.1.2 Historias de Usuario

Durante la fase de exploración se identificaron 22 Historias de Usuario (HU), representando cada una las funcionalidades del sistema, a continuación se ilustran algunas de las HU identificadas las cuales se describen en la siguientes figuras.

Historia de Usuario	
Número: *HU_1*	**Nombre Historia de Usuario**: Autenticar usuario.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Roberto Rielo Villadóniga	**Iteración Asignada**: 1
Prioridad en Negocio: Alta	**Puntos Estimados**: 1.
Tipo de actividad: Nueva	**Puntos Reales**: 1
Descripción: El usuario accede al sistema logiandose mediante un nombre de usuario y su contraseña en el dominio uci, el sistema comprobará que estos datos son correctos, al serlo, el sistema le dará acceso al usuario a las funcionalidades a las que tiene permiso.	
Observaciones: Todos los usuarios deben autenticarse para acceder a la aplicación.	
RF# 1	

Ilustración 2: Descripción de la HU Autenticar usuario.

Historia de Usuario	
Número: *HU_2*	**Nombre Historia de Usuario**: Asignar área a la queja
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Carlos J. Pérez de la Cruz	**Iteración Asignada**: 1
Prioridad en Negocio: Alta	**Puntos Estimados**: 1
Tipo de actividad: Nueva	**Puntos Reales**: 1
Descripción: Se le asigna un área a la queja para la cual será enviada y dónde el responsable de esa área le dará seguimiento.	
Observaciones: El usuario debe estar previamente autenticado en el sistema, se deben haber consultado las quejas previamente con la decana y la decana debe haber autorizado la asignación del área que se le será asignada.	
RF# 2	

Ilustración 3: Descripción de la HU Asignar área a la queja.

Historia de Usuario	
Número: *HU_3*	**Nombre Historia de Usuario**: Asignar tiempo de respuesta de la queja
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Carlos J. Pérez de la Cruz	**Iteración Asignada**: 1
Prioridad en Negocio: Alta.	**Puntos Estimados**: 1
Tipo de actividad: Nueva	**Puntos Reales**: 1
Descripción: Se le define un término de tiempo a la queja en el cual el responsable del área a la que se envía tiene que darle respuesta.	
Observaciones: El usuario debe estar previamente autenticado en el sistema, se deben haber consultado las quejas previamente con la decana y la decana debe haber autorizado la asignación del tiempo de respuesta.	
RF# 3	

Ilustración 4: Descripción de la HU Asignar tiempo de respuesta de la queja.

Historia de Usuario	
Número: *HU_4*	**Nombre Historia de Usuario**: Gestionar queja
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Carlos J. Pérez de la Cruz	**Iteración Asignada**: 1
Prioridad en Negocio: Alta	**Puntos Estimados**: 2
Tipo de actividad: Nueva	**Puntos Reales**: 2
Descripción: Permite crear, modificar, eliminar, actualizar y listar las quejas.	
Observaciones: El usuario debe estar previamente autenticado en el sistema. Para crear y modificar una queja al usuario se le mostrará un editor de texto donde podrá escribir la nueva queja o modificar la existente.	
RF# 4; RF# 5; RF# 6; RF# 7; RF# 8	

Ilustración 5: Descripción de la HU Gestionar queja.

Historia de Usuario	
Número: *HU_6*	**Nombre Historia de Usuario**: Gestionar organigrama de la facultad
Modificación de Historia de Usuario Número: Ninguna.	
Usuario: Roberto Rielo Villadóniga	**Iteración Asignada**: 1
Prioridad en Negocio: Alta	**Puntos Estimados**: 2
Tipo de actividad: Nueva	**Puntos Reales**: 2
Descripción: Permite modificar y visualizar la estructura jerárquica de la facultad, donde los usuarios luego de conocer a que área le fue asignada cada una de sus quejas podrán ver quién es el encargado de darle seguimiento y respuesta a la misma.	
Observaciones: El usuario debe estar previamente autenticado en el sistema. Solo podrá modificarlo y actualizarlo el usuario con los permisos requeridos para esta acción.	
RF# 11 ; RF# 12	

Ilustración 6: Descripción de la HU Gestionar organigrama de la facultad.

2.1.3 Plan de Iteraciones

En este plan se definen varias iteraciones sobre el sistema antes de ser entregado. Algunos de los elementos que deben tenerse en cuenta para su elaboración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas y tareas no terminadas en la iteración. El plan especifica exactamente que historias de usuarios serán implementadas para cada lanzamiento del sistema. Esto da al cliente un grupo de historias de usuario para incluir en el próximo plan de iteración. (43)

A continuación se muestra el plan de iteraciones definido para el desarrollo del sistema:

Tabla 3: Descripción del plan de iteraciones.

Número de Iteración	Descripción de la iteración	HU a implementar
1	Se tuvo en cuenta aquellas historias de usuario de mayor importancia en cuanto a la funcionalidad que describen cada una, que se ve reflejado en la alta prioridad para el negocio.	**HU_1:** Autenticar usuario **HU_2:** Asignar área a la queja **HU_3:** Asignar tiempo de tramitación a la queja **HU_4:** Gestionar queja **HU_6:** Gestionar organigrama de la facultad **HU_10:** Mostrar perfil personal **HU_11:** Gestionar nomencladores **HU_12:** Ver detalles de queja

		HU_17: Gestionar estudiante por grupo **HU_18:** Gestionar estudiante por Comité de Base **HU_19:** Gestionar trabajador por Comité de Base **HU_20:** Gestionar Trabajador por Factor
2	Se tuvo en cuenta aquellas funcionalidades del sistema con menos prioridad, es decir, aquellas historias de usuario que presentan prioridad media o baja. Por otra parte en esta iteración se corregirán los errores encontrados en la iteración anterior, obteniéndose una nueva versión del sistema.	**HU_5**: Consultar quejas por categorías **HU_7:** Consultar quejas generadas por factor **HU_8:** Consultar quejas por estado **HU_9:** Consultar quejas asignadas por áreas **HU_13:** Generar alerta**s** **HU_14:** Consultar quejas generadas por áreas **HU_15:** Reporte de estado de las quejas asignadas por áreas y factores **HU_16:** Consultar quejas asignadas por factor

2.1.4 Plan de Entregas

El plan de entregas es un documento que especifica exactamente que historias de usuario serán implementadas en cada entrega del sistema y sus prioridades, de modo que también permita conocer con exactitud qué historias de usuario serán implementadas en la próxima liberación. La idea principal del plan es hacer entregas frecuentes para obtener una mayor retroalimentación. A continuación se presenta una propuesta para el plan de entrega elaborado por el equipo de desarrollo donde se reflejan las fechas de entregas para las primeras versiones de las historias de usuario.

Tabla 4: Descripción del plan de entregas.

Número de iteración	HU a implementar	Fecha de entrega
1	**HU_1:** Autenticar usuario **HU_2:** Asignar área a la queja **HU_3:** Asignar tiempo de tramitación a la queja **HU_4:** Gestionar queja **HU_6:** Gestionar organigrama de la facultad **HU_10:** Mostrar perfil personal **HU_11:** Gestionar nomencladores **HU_12:** Ver detalles de queja **HU_17:** Gestionar estudiante por grupo **HU_18:** Gestionar estudiante por Comité de Base	24 de abril de 2015

	HU_19: Gestionar trabajador por Comité de Base **HU_20:** Gestionar Trabajador por Factor	
2	**HU_5**: Consultar quejas por categorías **HU_7:** Consultar quejas generadas por factor **HU_8:** Consultar quejas por estado **HU_9:** Consultar quejas asignadas por áreas **HU_13:** Generar alerta**s** **HU_14:** Consultar quejas generadas por áreas **HU_15:** Reporte de estado de las quejas asignadas por áreas y factores **HU_16:** Consultar quejas asignadas por factor	25 de mayo de 2015

2.2 Diseño

XP sugiere que hay que conseguir diseños simples y sencillos. Para procurar hacerlo todo lo menos complicado posible para el usuario o cliente y conseguir un diseño fácilmente entendible que costará menos tiempo y esfuerzo para desarrollarlo.

2.2.1 Tarjetas CRC

Las tarjetas CRC, Class, Responsibilities and Collaboration (CRC por sus siglas en inglés) sirven para diseñar el sistema en conjunto entre todo el equipo. Estas tarjetas representan objetos, para los cuales se especifica la clase a la que pertenece dicho objeto, las responsabilidades u objetivos que debe cumplir y las clases que colaboran con cada responsabilidad.

A continuación se muestran algunas de las tarjetas CRC creadas.

Tabla 5: Tarjeta CRC Clase: Autenticación.

Responsabilidad	Clases relacionadas
➢ Verificar si un usuario pertenece a la facultad 3	Default

Tabla 6: Tarjeta CRC Clase: Organigrama.

Responsabilidad	Clases relacionadas
➢ Visualizar el organigrama de la facultad ➢ Modificar el organigrama de la facultad ➢ Añadir los roles a los usuarios	Organigrama Rol Usuario

Tabla 7: Tarjeta CRC Clase: Queja.

Responsabilidad	Clases relacionadas
➢ Visualizar el listado de las quejas así como su estado y el área a la que son asignadas ➢ Asignarle un área a la queja y un tiempo de tramitación ➢ Añadir una nueva queja al sistema ➢ Eliminar una queja ➢ Modificar una queja	Queja Rol Usuario AreaFacultad

2.2.2 Patrón de Arquitectura

Para estructurar el desarrollo del software se usó como patrón arquitectónico el MVC apoyados en los principios y arquitectura que propone el marco de trabajo Symfony2. En el desarrollo del sistema se define una arquitectura en capas basada en el patrón MVC, debido a las facilidades que ofrece tanto en términos de escalabilidad, pues divide la lógica del negocio de la lógica del diseño como en la sencillez de mantenimiento, además de mantener un bajo acoplamiento. Este patrón se muestra de forma implícita en el uso del marco de trabajo Symfony 2, definiendo este último dónde se ubican las clases del modelo, las de la vista y las del controlador.

En la siguiente figura se muestra como queda estructurado el sistema con la aplicación del patrón MVC.

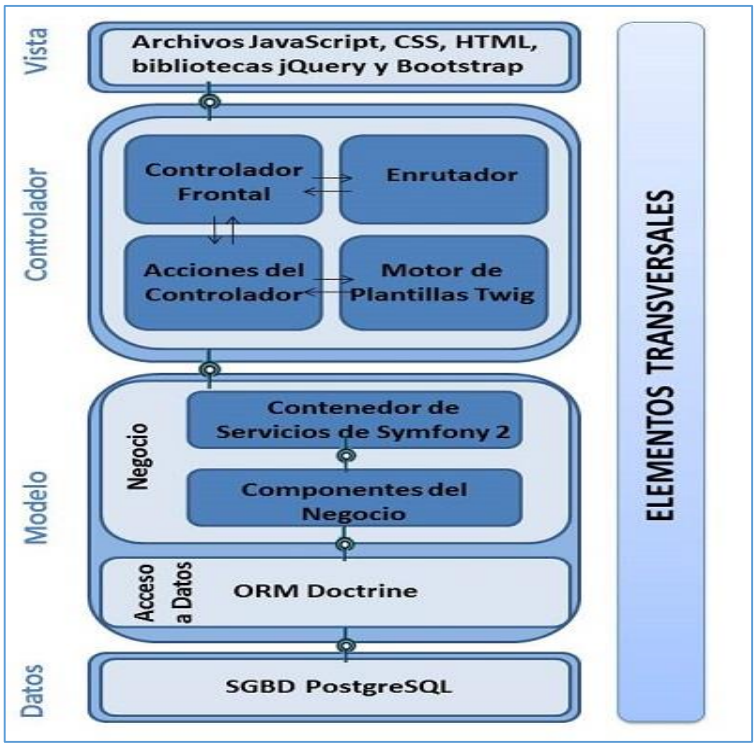


Ilustración 7: Arquitectura del sistema.

A continuación se muestra como quedan organizados los archivos del sistema con la aplicación de este patrón.

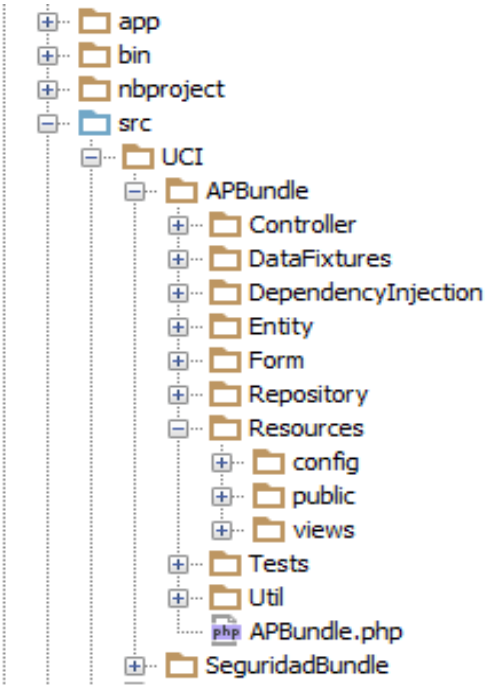


Ilustración 8: Organización de los archivos del sistema.

Capa controladora: los controladores se encuentran ubicados en la carpeta Controller como muestra la ilustración 9 donde se recogen todas las clases controladoras implementadas en el desarrollo del sistema:

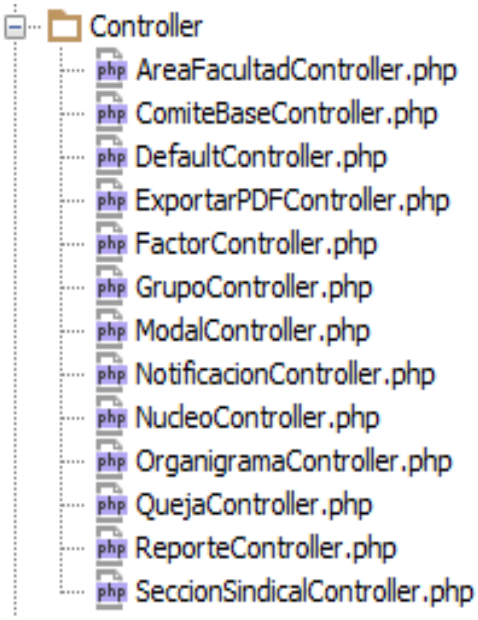


Ilustración 9: Carpeta contenedora de las clases controladoras

Capa vista: la implementación de esta capa se encuentra ubicada en el paquete Resources, específicamente en views, donde se encuentran todas las páginas de la aplicación, las cuales importan las imágenes, archivos CSS y JavaScript necesarios, como lo muestra la figura 10.

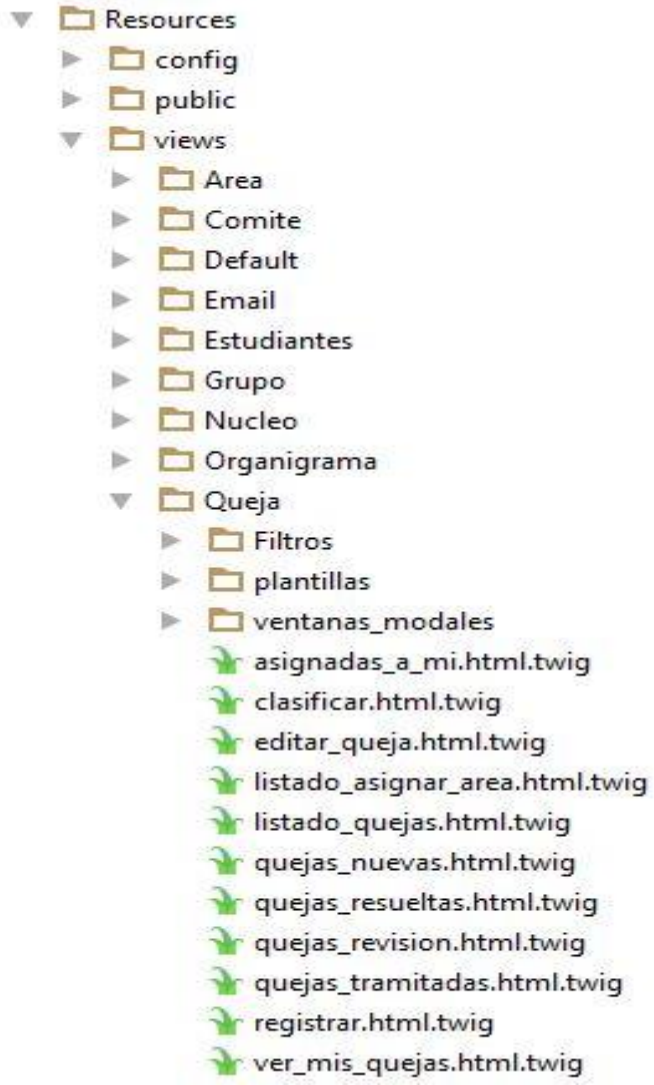


Ilustración 10: Carpeta contenedora de las vistas.

Capa del modelo: está compuesta por los componentes del negocio, donde se encuentran las entidades y los repositorios, encargadas del almacenamiento, recuperación e inserción de los datos y ubicadas en la carpeta Entity como muestra la figura 11.

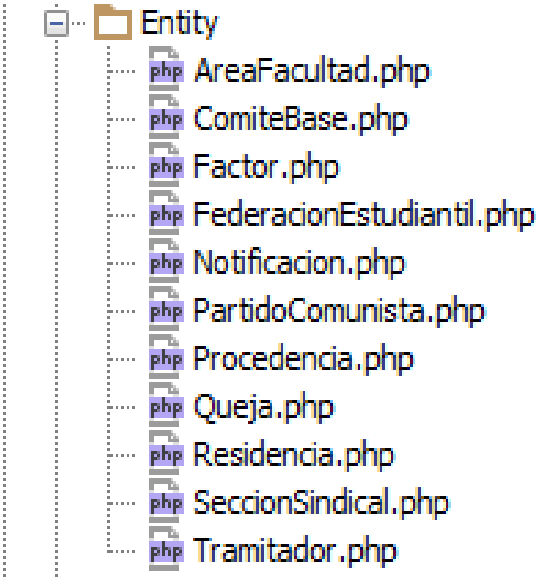


Ilustración 11: Carpeta contenedora de las entidades del negocio.

Componentes verticales:

Seguridad: la autenticación se realiza a través del servicio Ldap, lo que disminuye la posibilidad de que exista un ataque al sistema y se puedan obtener las contraseñas de los usuarios. Mediante el control de acceso basado en roles se gestionan los permisos a las distintas funcionalidades para cada uno de los usuarios que acceden al sistema, teniendo en cuenta los principios de seguridad del mínimo privilegio, integridad y confidencialidad. Mediante la implementación de listas de control de acceso (ACL) se realiza la asignación de permisos específicos a cada objeto y para cada usuario de la aplicación.

Validaciones: se realizan tanto del lado del cliente como del lado del servidor, para garantizar la integridad y veracidad de los datos de acuerdo a las restricciones del negocio, garantizando un correcto funcionamiento del sistema.

En la ilustración 12 se muestra parte del código de una validación a la hora de insertar un grupo.

```
function validacion() {

    var grupo = $('#nuevo_grupo').val().trim();
    for (var i = 0; i < grupo.length; i++) {
        if (isNaN(grupo[i])) {
            $('#texto_error').text('No se acepta el caracter "' + grupo[i] + '".');
            $('#mensaje_error').attr('style', 'display:block');
            return false;
        }
    }
```

Ilustración 12: Realización de validaciones.

Tratamiento de excepciones: este elemento es uno de los más importantes en la implementación del sistema pues orienta al usuario ante la ocurrencia de algún error, se encuentra de forma transversal en toda la aplicación pues en cualquiera de las capas se puede lanzar una excepción. En el desarrollo del sistema lo que se realizó fue un tratamiento personalizado de excepciones para cada uno de los posibles errores que se pueden presentar acorde al negocio.

```
public function DevolverImageneDirigentes(Request $request)
{
    $datos = array();
    $rol_sistema = $request->request->get('rol');

    $em = $this->getDoctrine()->getManager();
    $rol = $em->getRepository('SeguridadBundle:Rol')->findOneBy(array('nombre' => $rol_sistema));
    if (!$rol) {
        throw new \ErrorException("No se ha encontrado ese rol " . $rol_sistema . " en la base de datos");
    }
    $usuario = $em->getRepository('SeguridadBundle:Usuario')->findOneBy(array('rol' => $rol));
    if (!$usuario) {
        throw new \ErrorException("No se ha encontrado ese usuario " . $usuario . " en la base de datos");
    }

    $datos[$rol_sistema] = array('usuario' => $usuario->getUsername(), 'foto' => $usuario->getFoto());

    return new JsonResponse($datos);
}
```

Ilustración 13: Tratamiento de errores

2.2.3 Patrones de Diseño

Para garantizar un mejor diseño del sistema propuesto se hizo uso de los siguientes patrones de diseños.

Patrones GRASP utilizados en la solución

Patrones generales de software para asignar responsabilidades.

Experto

Se basa en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. (39)

Algunas ventajas ofrecidas por el patrón son: (39)

- Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases "sencillas y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Este patrón se evidencia en el desarrollo del sistema en la creación de las vistas que serán mostradas al usuario, siendo las clases Controller las expertas en información, y por tanto, las encargadas de gestionar toda información necesaria para la generación de la vista que será mostrada al usuario.

A continuación se muestra en la ilustración 8 como la clase QuejaController se encarga de suministrar la información que será mostrada en la vista modal_detalles_quejas_asignadas.

```
public function modal_detalles_quejas_asignadasAction(Queja $queja)
{
    $em = $this->getDoctrine()->getManager();
    $tramitadores = $em->getRepository('APBundle:Queja')->ObtenerDatosTramitadoresDadoQueja($queja);
    $responsable = $em->getRepository('APBundle:Tramitador')->ObtenerResponsableDadoQueja($queja);
    $datos_usuario = $this->get("ap.recogerdatos")->DevolverUsuarioLogueado();

    return $this->render("@AP/Queja/ventanas_modales/modal_detalles_queja_asignadas", array(
        'queja' => $queja,
        'tramitadores' => $tramitadores,
        'responsable' => $responsable,
        'es_responsable' => $responsable === $datos_usuario
    ));
}
```

Ilustración 14: Uso patrón experto.

Creador

El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento, constituyendo su principal beneficio, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. (39)

Beneficios ofrecidos por el patrón (39):

- Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

Este patrón se evidencia en las clases controladoras que son las encargadas de crear las instancias de los objetos que manejan, favoreciendo así la reutilización y el bajo acoplamiento.
En la ilustración 9 se muestra parte del código fuente utilizado para la creación de una queja, evidenciándose el uso de este patrón en la creación del formulario a utilizar para recoger los datos de la nueva queja.

```
public function registrarAction(Request $request)
{
    $em = $this->getDoctrine()->getManager();
    $queja = new Queja();
    $formulario = $this->createForm(new QuejaType(), $queja);
    $datos_usuario = $this->get("ap.recogerdatos")->DevolverUsuarioLogueado();

    if ($request->isMethod('POST')) {
        $formulario->handleRequest($request);

        if ($formulario->isValid()) {
            $queja->setEmisor($datos_usuario);
            $em->persist($queja);
            $em->flush();
        }
        return $this->redirect($this->generateUrl('/queja/mis_quejas'));
    }
    return $this->render("@AP/Queja/registrar", array(
        'formulario' => $formulario->createView(),
        'usuario_logueado' => $datos_usuario
    ));
```

Ilustración 15: Uso patrón creador.

Bajo acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. El Bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como experto o alta cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades. (39)
Ventajas del uso del patrón (39):

- No se afectan por cambios de otros componentes.
- Fáciles de entender por separado.

- Fáciles de reutilizar.

Estas ventajas permitieron potenciar la reutilización y disminuir la dependencia entre las clases. En el desarrollo del sistema este patrón se evidencia en las entidades que son las clases más reutilizadas y son totalmente independientes, pues no están asociadas ni a la vista ni al controlador.

Alta cohesión

En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. (39)

Grady Booch señala que se da una alta cohesión funcional cuando los elementos de un componente (clase, por ejemplo) colaboran para producir algún comportamiento bien definido.

Entre los beneficios ofrecidos por el patrón se encuentran (39):

- Mejoran la claridad y la facilidad con que se entiende el diseño.
- Se simplifican el mantenimiento y las mejoras en funcionalidad.
- A menudo se genera un bajo acoplamiento.
- La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico.

Este patrón se evidencia en cada clase que realiza una labor única dentro del sistema y colabora con las otras para llevar a cabo una tarea, básicamente en las clases que forman parte de las capas controladora y modelo.

Patrones de Comportamiento (Patrones GOF):

Observador

Este patrón define una dependencia “uno-a-muchos” entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente. (44)

El patrón Observador se evidencia en la implementación de un Listener que se encarga de ejecutar la lógica asociada a una funcionalidad del negocio. Una vez que el componente Event Dispatcher del marco de trabajo Symfony2, quien es responsable de lanzar diferentes eventos durante la ejecución de una petición del usuario lanza el evento OnKernelRequest, este es recibido por una clase que observa el comportamiento de un objeto de tipo Request para introducir modificaciones a sus características.

En la aplicación se evidencia a la hora de recuperar la información perteneciente a la cantidad de quejas por estado y las notificaciones asociadas a las respuestas dadas al usuario, la cual es necesario que esté actualizada luego de cada acción ejecutada en el sistema.

Decorador

Añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. Provee una alternativa muy flexible para agregar funcionalidad a una clase. (44)

En la implementación del sistema el patrón decorador se evidencia en la potencialidad que posee el motor de plantillas Twig relacionado con la herencia entre plantillas, es decir, en el uso de una plantilla global que contendrá los elementos comunes del sitio para las vistas que decorarán las demás páginas de la aplicación.

```
{% extends "::base.html.twig" %}

{% block titulo %} Registrar {% endblock %}
```

Ilustración 16: Uso patrón decorador

Patrones de diseño de base de datos

Llaves subrogadas

Este patrón es muy utilizado pues se decide generar una llave primara única para cada entidad en vez de usar un atributo identificador en el contexto dado. Normalmente se usa enteros en columnas identity o GUID (Global Unique Identifier) que están demostrados que no se repiten o con una probabilidad extremadamente baja. Permite que las tablas sean más fáciles de consultar por el identificador dado que se conoce el mismo tipo de todos en cada tabla. (45)

El uso de este patrón constituye una protección ante los cambios debido a que la lógica del negocio no está en las llaves y evita la contención (bloqueo) de la base de datos, debido a la rapidez de los mecanismos de generación que provee el sistema.

Ilustración 17: Uso del patrón Llaves Subrogadas en la entidad Usuario

Control de Acceso Basado en Roles (RBAC, por sus siglas en inglés): este patrón se implementa mediante la asignación de roles a los diferentes usuarios de la aplicación. Cada rol posee permisos para realizar determinadas funciones de acuerdo a las restricciones del negocio. Por lo que cada usuario tiene estrictamente los permisos que les son conferidos a través del rol que desempeña.

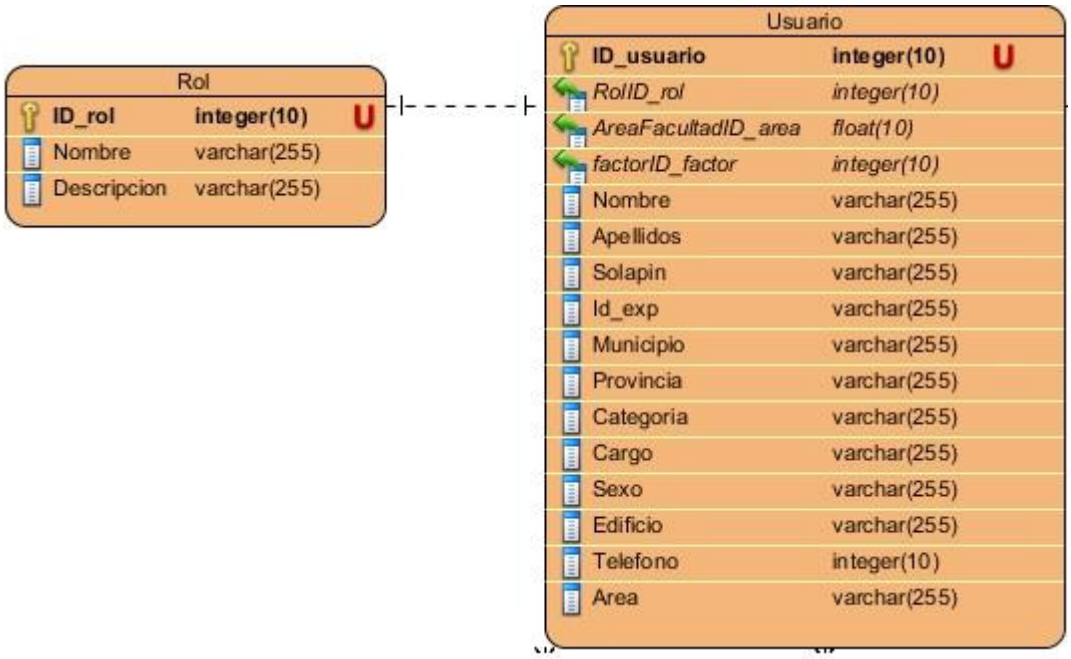


Ilustración 18: Uso del patrón RBAC.

2.2.4 Diagrama de Clases del Diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información (39):

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de atributos.
- Navegabilidad y dependencias.

A continuación se muestra una vista estática del diagrama de clases del diseño correspondiente a la HU Gestionar Queja donde se muestra la clase controladora y sus relaciones con los diferentes componentes que forman partes del diseño (vistas, métodos, navegabilidad, clases).

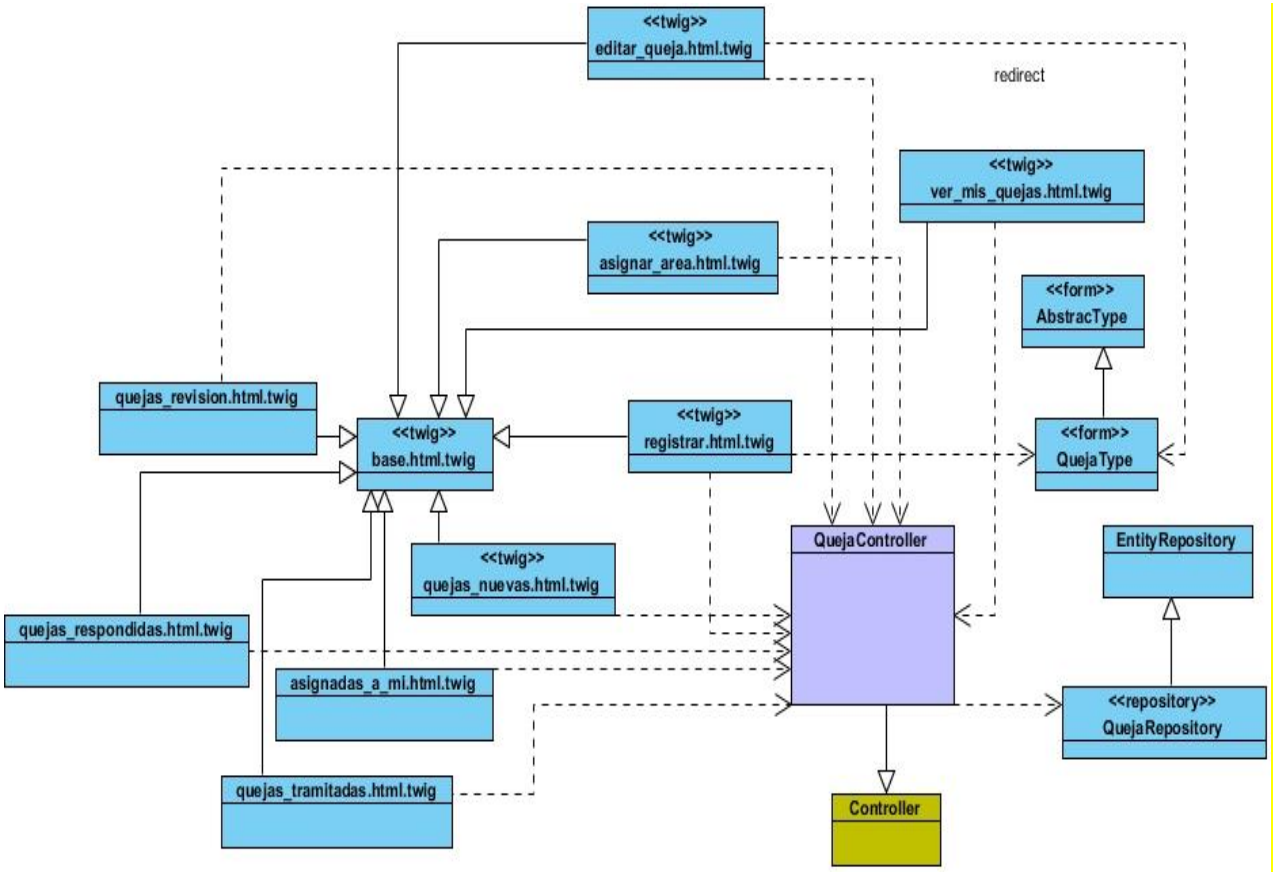


Ilustración 19: Diagrama de clase del diseño de la HU Gestionar queja.

2.2.5 Modelo de Datos

Conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Los modelos de datos comprenden aspectos relacionados con: estructuras y tipos de datos, operaciones y restricciones. (46)

El modelo de datos es un conjunto de conceptos que sirven para describir la estructura, semántica y las relaciones existen entre las entidades de una base de datos importantes para el funcionamiento del negocio y muestra los datos que serán contenidos en el sistema. (47)

A continuación se presenta el modelo de datos a partir del cual se definió la base de datos usada en el sistema, el mismo se muestran las 11 tablas que lo conforman y sus relaciones.

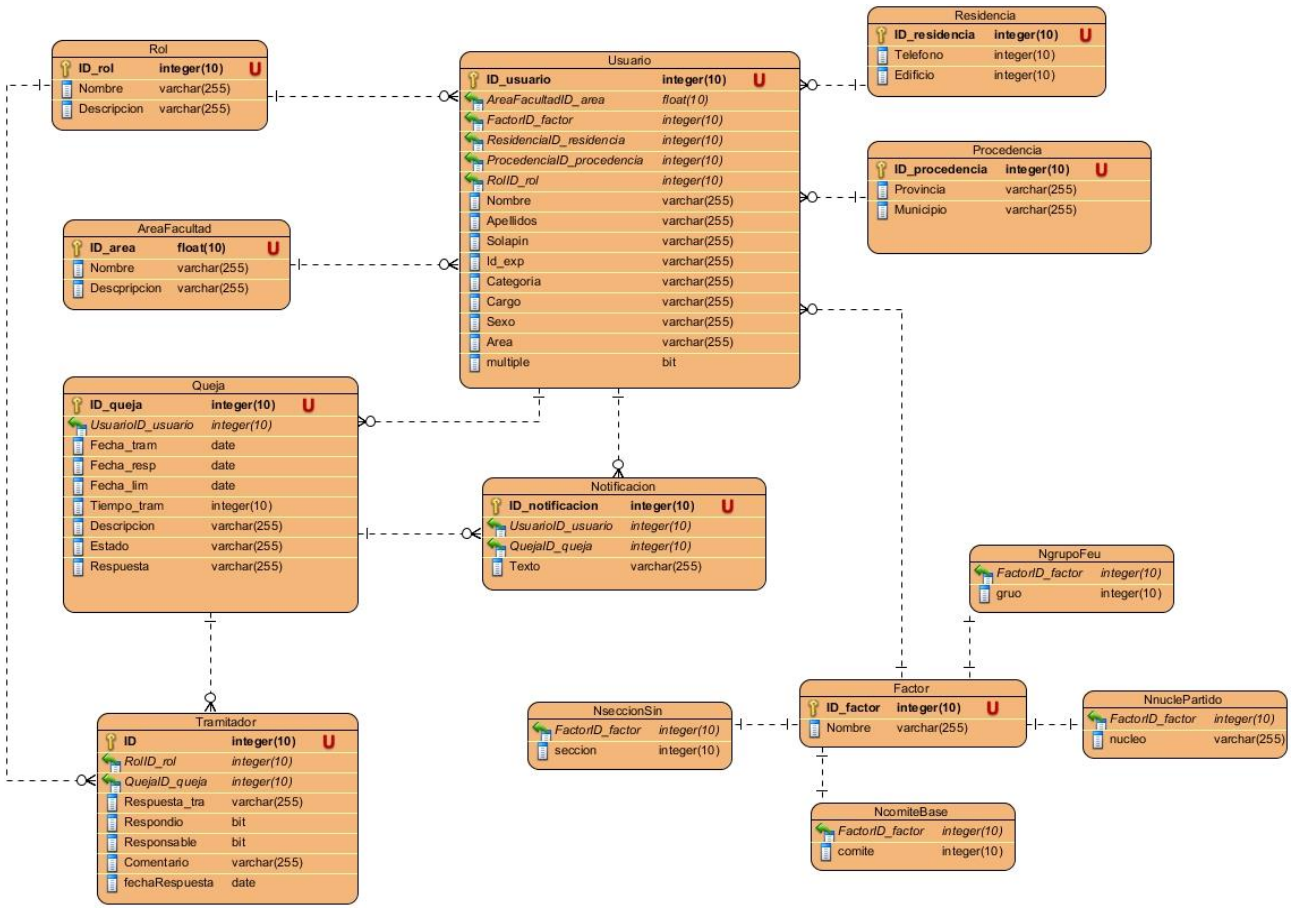


Ilustración 20: Modelo de datos.

2.3 Implementación

2.3.1 Estándares de Codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento.

Con el objetivo de lograr una estandarización en la programación de la aplicación web se decide aplicar los estándares de codificación en PHP (PSR0, PSR1, PSR2 y PSR3) (48).

PSR-0

Este estándar especifica las siguientes reglas:

- Los namespaces y las clases deben tener la siguiente estructura <Vendor name>(<Namespace>)*<Class Name>
- Cada namespace debe tener un namespace superior ("Vendor name").
- Cada namespace puede tener tantos sub-namespaces como se quiera.
- Los nombres de los namespaces o clases deben ser separados por un guion bajo (_).

- Todos los archivos deben tener la extensión .php.
- Los nombres de los namespaces o clases deben ser ordenadas alfabéticamente.

PSR-1, PSR-2 y PSR3

- Los archivos deben utilizar solamente <?php y las short tags <?=
- Los nombres de espacio y las clases deben seguir las reglas del PSR-0.
- Los nombres de clases deben ser escritas utilizando la técnica StudlyCaps[9].
- Las constantes deben ser definidas en MAYÚSCULAS y utilizando guion bajo (_) cómo separador.
- Los métodos y funciones deben ser escritos utilizando la técnica camelCase[10].
- Las llaves deben de estar abajo solamente en las clases y métodos.
- Las constantes true, false y null deben ser escritos en minúsculas.
- Debe haber un espacio después de cada estructura de control (if, for, foreach, while, switch, try...catch, etc.).

2.3.2 Tareas de la Ingeniería

Luego de la descripción de las historias de usuarios del sistema, se hace necesario la definición un grupo de tareas ingenieriles orientadas a dar cumplimiento a la implementación de las funcionalidades definidas en ellas, lo cual sienta las bases para el posterior desarrollo de la solución. Las tareas de ingeniería son un conjunto de acciones a desarrollar para resolver las historias de usuario. Permiten organizar el proceso de implementación además de posibilitar que sea conocido el grado de complejidad de cada historia de usuario teniendo en cuenta la cantidad de tareas asociadas a ella.

Tabla 8: Descripción de la tarea de la ingeniería de la HU-1.

Tareas de la Ingeniería	
Número de Tarea: T_4	**Número de Historia de Usuario:** HU-1
Nombre Tarea: Diseñar las interfaces requeridas para la funcionalidad Autenticar usuario	
Tipo Tarea: Desarrollo	**Puntos Estimados (semanas):** 1
Fecha Inicio:2-2-2015	**Fecha Fin:6-2-2015**
Programador Responsable: Roberto y Carlos	
Descripción: Se crean las interfaces necesarias	

[9] Studlycaps es una forma de notación de texto en el que la capitalización de letras varía según algún patrón, o arbitrariamente, por lo general también omitiendo espacios entre palabras y a menudo omitiendo algunas letras.
[10] CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas
El nombre CamelCase se podría traducir como Mayúsculas/Minúsculas Camello

Tabla 9: Descripción de la tarea de la ingeniería de la HU-1.

Tareas de la Ingeniería	
Número de Tarea: T_5	**Número de Historia de Usuario:** HU-1
Nombre Tarea: Implementar funcionalidad: Autenticar usuario	
Tipo Tarea: Desarrollo	**Puntos Estimados (semanas):** 1
Fecha Inicio:2-2-2015	**Fecha Fin:6-2-2015**
Programador Responsable: Roberto y Carlos	
Descripción: Se crean las clases y métodos necesarios para la implementación de la funcionalidad	

Tabla 10: Descripción de la tarea de la ingeniería de la HU-2.

Tareas de la Ingeniería	
Número de Tarea: T_6	**Número de Historia de Usuario:** HU-2
Nombre Tarea: Diseñar las interfaces requeridas para la funcionalidad: Asignar área a la queja.	
Tipo Tarea: Desarrollo	**Puntos Estimados (semanas):** 1
Fecha Inicio:20-4-2015	**Fecha Fin:25-4-2015**
Programador Responsable: Roberto y Carlos	
Descripción: Se crean las interfaces necesarias	

Tabla 11: Descripción de la tarea de la ingeniería de la HU-2.

Tareas de la Ingeniería	
Número de Tarea: T_7	**Número de Historia de Usuario:** HU-2
Nombre Tarea: Implementar funcionalidad: Asignar área a la queja	
Tipo Tarea: Desarrollo	**Puntos Estimados (semanas):** 1
Fecha Inicio:20-4-2015	**Fecha Fin:25-4-2015**
Programador Responsable: Roberto y Carlos	
Descripción: Se crean las clases y métodos necesarios para la implementación de la funcionalidad	

Tabla 12: Descripción de la tarea de la ingeniería de la HU-4.

Tareas de la Ingeniería	
Número de Tarea: T_10	**Número de Historia de Usuario:** HU-4
Nombre Tarea: Diseñar las interfaces requeridas para la funcionalidad: Gestionar queja (crear, modificar, eliminar, actualizar, listar).	
Tipo Tarea: Desarrollo	**Puntos Estimados (semanas):** 3
Fecha Inicio:2-3-2015	**Fecha Fin:18-3-2105**
Programador Responsable: Roberto y Carlos	
Descripción: Se crean las interfaces necesarias para cada uno los elementos del gestionar teniendo en cuenta los datos que se deben mostrar en cada uno de ellos	

Tabla 13: Descripción de la tarea de la ingeniería de la HU-4.

Tareas de la Ingeniería	
Número de Tarea: T_ 11	**Número de Historia de Usuario:** HU-4
Nombre Tarea: Implementar funcionalidad: Gestionar queja (crear, modificar, eliminar, actualizar, listar).	
Tipo Tarea: Desarrollo	**Puntos Estimados (semanas):** 3

Fecha Inicio: 2-3-2015	Fecha Fin:18-3-2015
Programador Responsable: Roberto y Carlos	
Descripción: Se crean las clases y los métodos necesarios para implementar cada una de las funcionalidades	

2.3.3 Diagrama de Componentes

Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema. En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. A continuación se muestra el diagrama de componentes perteneciente al sistema.

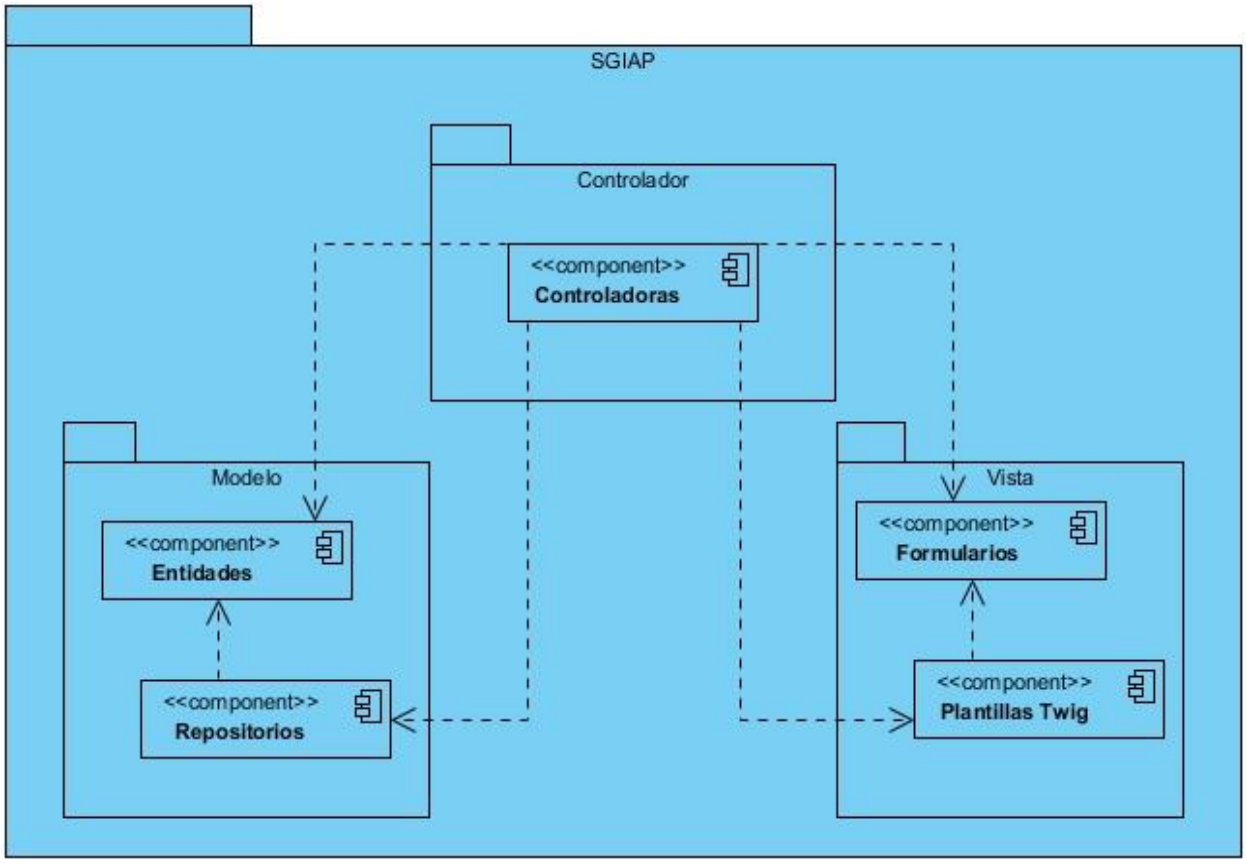


Ilustración 21: Diagrama de componentes.

2.3.4 Diagrama de Despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware y muestra cómo los elementos y artefactos del software se trazan en esos elementos de hardware. (49)

A continuación se muestra el diagrama de despliegue para el sistema propuesto, el mismo está compuesto por una PC[11] cliente conectada a través del protocolo HTTPS a un servidor web, que se comunica con el servidor de bases de datos mediante el protocolo TCP/IP y con el servidor Ldap UCI para la autenticación de los usuarios mediante el protocolo HTTPS, además de una impresora que se conecta mediante USB a la PC cliente, esto ocurre en cada una de las instancias. El servidor

[11] Personal Computer

de bases de datos establece una conexión con el Centro de datos al cual envía toda la información que se procesa y a su vez obtiene aquella que necesite.

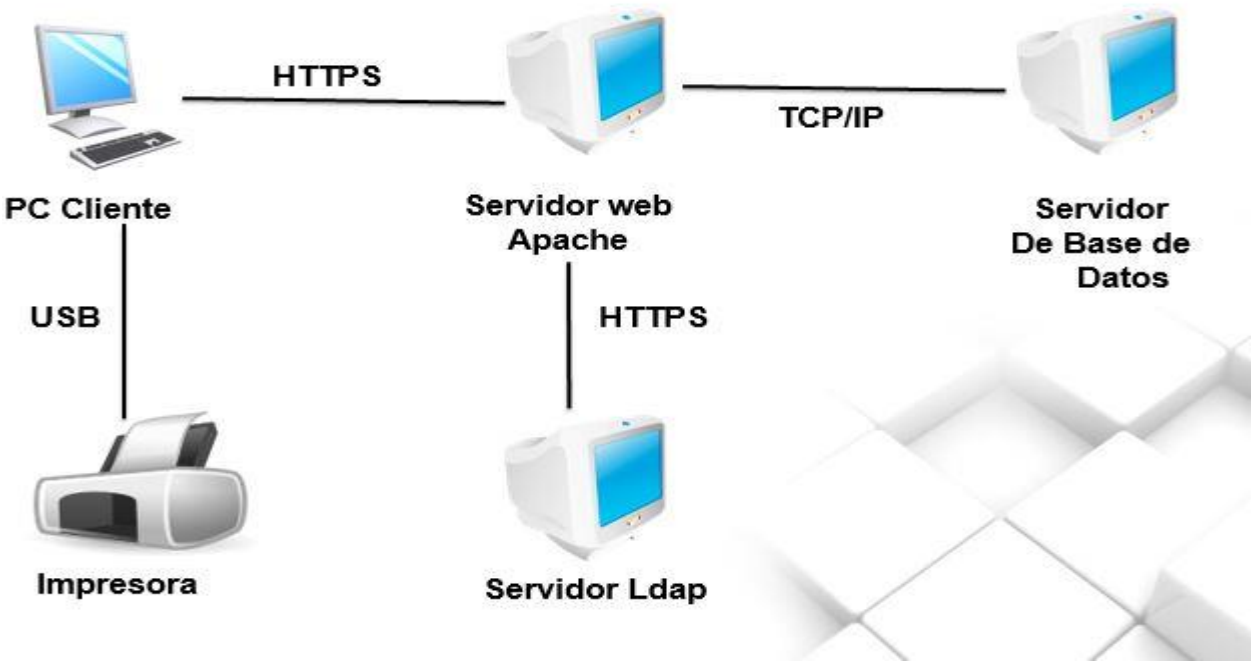


Ilustración 22: Diagrama de despliegue.

CONCLUSIONES DEL CAPÍTULO

Tras haber concluido con las tres primeras fases propuestas por la metodología se arribó a las siguientes conclusiones:

- Se elaboraron los artefactos correspondientes a los flujos de trabajo indicados por la metodología de desarrollo XP, creando así la documentación necesaria de la investigación que sirvió de guía a los desarrolladores para la implementación de la solución.
- Se trataron los aspectos más significativos del proceso de implementación, donde se mostró el Plan de Iteraciones y el estándar de codificación empleado en la solución lo que permitió realizar las diferentes entregas al cliente y estandarizar el código del sistema para la realización de futuros mantenimientos.
- Se obtuvo la implementación de la aplicación, dando solución a los requisitos funcionales identificados.

Capítulo 3. Validación y Pruebas

En el presente capítulo se muestran los resultados obtenidos de la aplicación de las métricas y técnicas empleadas para validar los requisitos y el diseño del sistema; además de los resultados alcanzados luego de la realización las pruebas de funcionalidad y unitarias que propone la metodología.

3.1 Validación

Dando cumplimiento a las técnicas de validación propuestas en el Capítulo 1 para validar los requisitos y el diseño propuesto a continuación se muestran los resultados obtenidos en cada una de las técnicas aplicadas.

3.1.1 Validación de Requisitos

Con el objetivo de verificar si los requisitos del software obtenidos definen el sistema que el cliente desea, se llevó a cabo un proceso de validación, para el cual se emplearon las técnicas propuestas en el Capítulo 1 obteniéndose el siguiente resultado para cada una de ellas:

Revisiones de requisitos: se realizaron revisiones a cada uno de los requisitos por parte del equipo de desarrollo conjuntamente con el cliente. En una primera revisión se generaron un total de 4 no conformidades, las cuales fueron corregidas satisfactoriamente y en tiempo. Con el cliente se desarrollaron 2 revisiones. En la primera, el mismo quedó satisfecho con el trabajo efectuado por el equipo de desarrollo, sin embargo se modificaron detalles a 6 de los requisitos funcionales para su perfeccionamiento y se añadieron 12 requisitos. En el segundo encuentro se aprobaron todos los requisitos, generándose de este encuentro el Acta de aceptación por parte del cliente.

Construcción de prototipos: mediante los prototipos se le mostró al cliente un modelo ejecutable del sistema que le permitió tener una visión preliminar del funcionamiento de este y a través de la interacción con las distintas funcionalidades se comprobó si satisfacía sus necesidades, los mismos fueron aprobados por el cliente con un alto nivel de satisfacción.

Métrica para Calidad de Especificación

Para medir la calidad de especificación de los requisitos se usó la métrica de Calidad de la especificación (CE), la cual permite obtener cuán entendibles y precisos son los requisitos (40).

Primeramente se calcula el total de requisitos de la especificación como se muestra a continuación:

Nr = Nf + Nnf
Nr = 61 + 21
Nr = 82

Donde:

Nr: *número de requisitos de especificación*.
Nf: *número de requisitos funcionales.*
Nnf: *número de requisitos no funcionales*.

Para determinar la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación: **ER = Nui / Nr.**

Donde ***Nui*** es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas.

ER = Nui / Nr
ER = 79/82
ER = 0.96

En una primera revisión se obtuvo un 0.96% de ambigüedad ya que se obtuvo por parte de los revisores una interpretación idéntica para 82 requisitos identificándose 3 requisitos ambiguos los cuales fueron modificados.

3.1.2 Validación del Diseño

Para comprobar la calidad y fiabilidad del diseño del sistema, se emplearon las métricas basadas en clases propuestas en el Capítulo 1: Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC), permitiendo el resultado de ambas métricas evaluar el diseño propuesto en la investigación. La aplicación detallada de las métricas se encuentra recogida a continuación donde se muestra un resumen de los resultados obtenidos.

Tamaño Operacional de Clases (TOC)

A continuación se muestra el resultado de la aplicación de la métrica TOC en las 22 clases que juegan un rol fundamental en la implementación de los principales procesos del sistema.

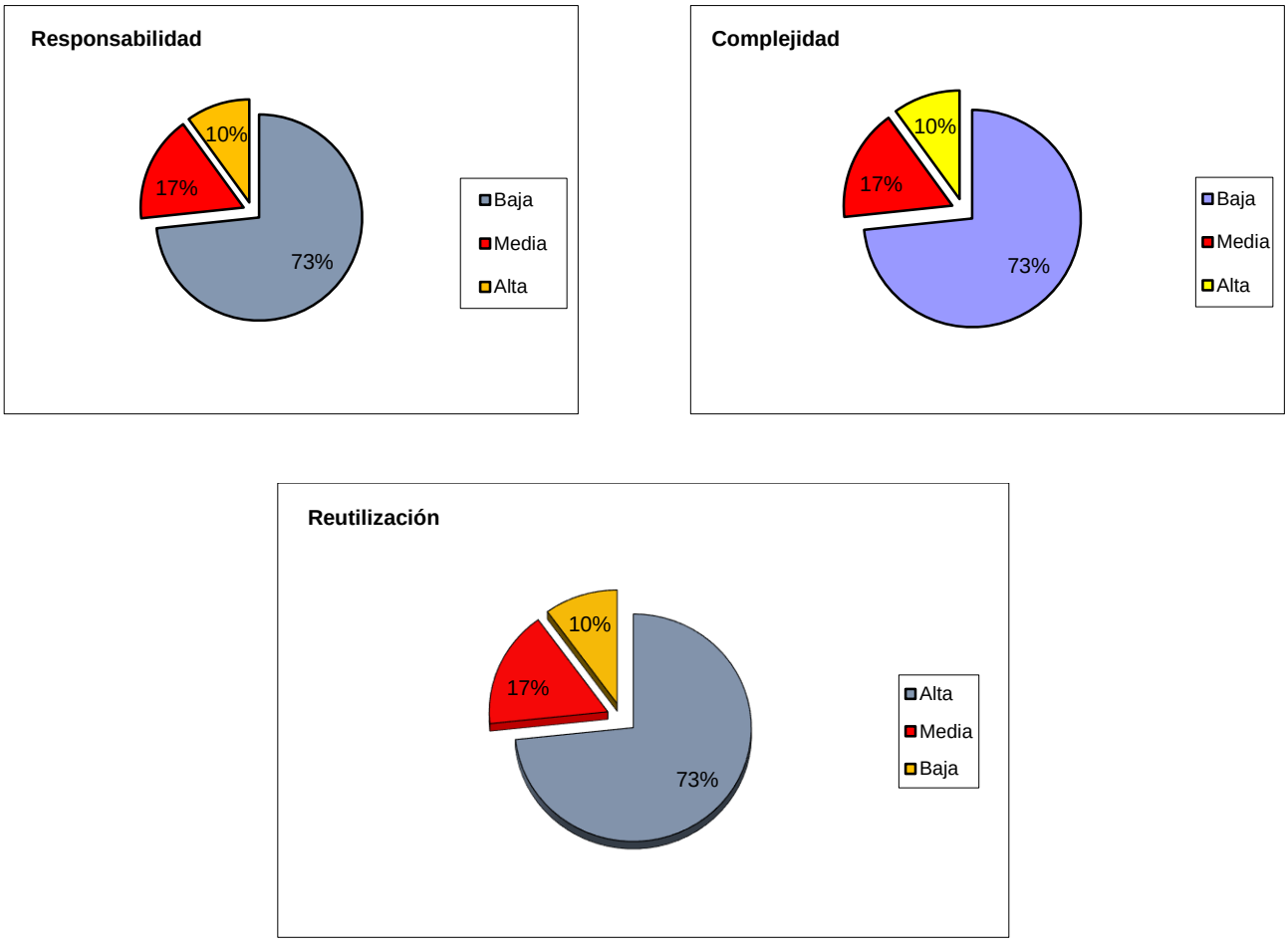


Ilustración 23: Resultados de la aplicación de la métrica TOC en los atributos de calidad Acoplamiento, Complejidad de Mantenimiento y Reutilización.

Relaciones entre Clases (RC)

A continuación se muestra el resultado de las medidas de los parámetros de calidad obtenidas luego de la aplicación de la métrica RC.

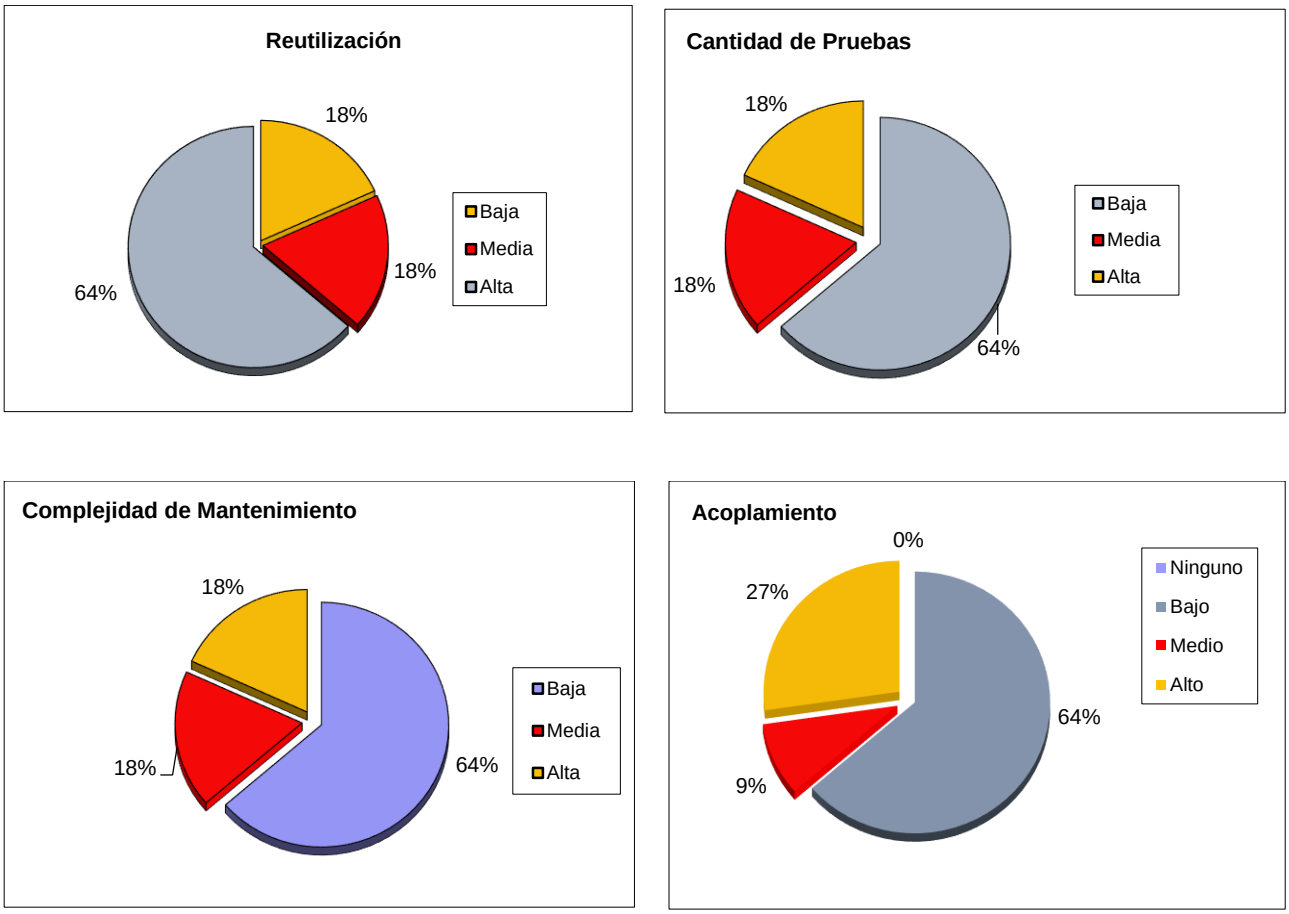


Ilustración 24: Resultados de la aplicación de la métrica RC en los atributos de calidad Acoplamiento, Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización.

A partir de los resultados obtenidos con la aplicación de las métricas se puede llegar a la conclusión de que el diseño presenta un acoplamiento relativamente bajo, la complejidad de mantenimiento se comporta de forma satisfactoria, pues aproximadamente el 64% de las clases son de fácil soporte y la mayoría de sus clases presentan un alto grado de reutilización.

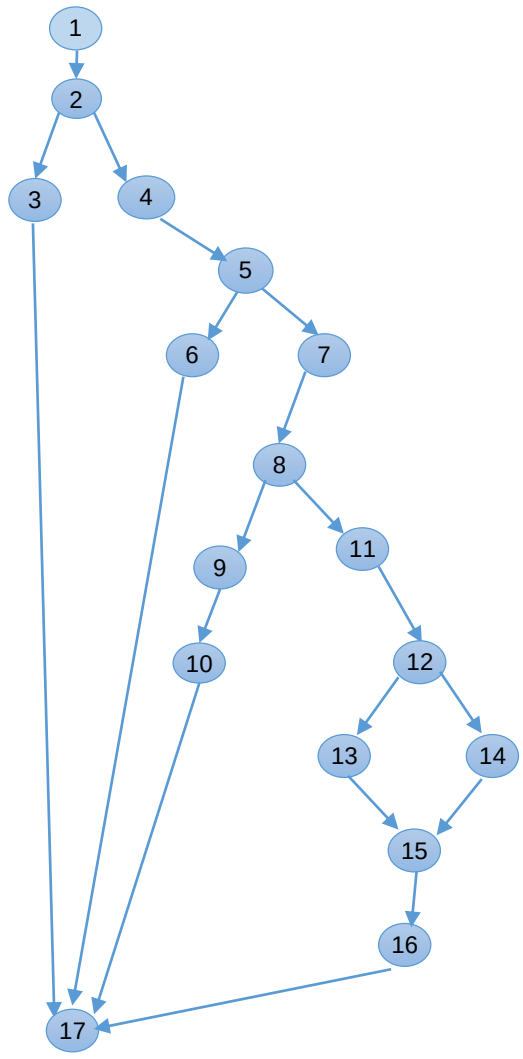
3.2 Pruebas

3.2.1 Pruebas Unitarias (caja blanca)

Para comprobar que cada sentencia de código se ejecuta al menos una vez, se realizaron pruebas al código de las funcionalidades más complejas desde el punto de vista de la programación en cada uno de los Bundles de la aplicación siguiendo la técnica propuesta en el Capítulo1.

Seguidamente se muestra el proceso de prueba realizado al método buscarUsuario (Ver Ilustración 25) perteneciente al Bundle APBundle, específicamente a la controladora OrganigramaController.

Grafo de flujo correspondiente al código del método.



La complejidad ciclomática será calculada mediante la siguiente fórmula:

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

Realizando los cálculos correspondientes se obtiene el siguiente resultado:

V (G) = 20-17+2

V (G) =5

Ilustración 25: Descripción de la aplicación de la Técnica del Camino Básico.

Por lo que el conjunto de caminos básicos sería:

Camino 1: 1-2-3-17

Camino 2: 1-2-4-5-6-17

Camino 3: 1-2-4-5-7-8-9-10-17

Camino 4: 1-2-4-5-7-8-11-12-13-15-16-17

Camino 5: 1-2-4-5-7-8-11-12-14-15-16-17

Por tanto existen 5 rutas independientes en el método analizado, por lo que serán necesarios igual número de casos de pruebas para verificar que se prueben todos los posibles datos de entradas en el método.

A continuación se muestra el resultado de las pruebas aplicadas a los caminos básicos 2 y 3

Tabla 14: Descripción del caso de prueba aplicado al camino básico 2.

Descripción: Se verificará que el sistema permita añadir una persona que existe en la base de datos y no tenga ningún puesto en el organigrama de la Facultad.		
Condición de Ejecución	**Entrada.**	**Resultados esperados**
La persona a ubicar debe existir en la base de datos de la aplicación.	Se busca un usuario.	El sistema debe añadir el usuario al organigrama y cambiarle el rol que el mismo tenía antiguamente.
Resultado: Satisfactorio.		

Tabla 15: Descripción del caso de prueba aplicado al camino básico 3.

Descripción: Se verificará que el sistema no permita añadir un usuario que y existe en la base de datos y tenga un lugar asociado en el organigrama.		
Condición de Ejecución	**Entrada.**	**Resultados esperados**
La persona a ubicar debe existir en base de datos y tener un lugar asociado en el organigrama de la Facultad.	Se busca un usuario.	El sistema debe mostrar una notificación de que el usuario buscado ya está asociado a uno de los puestos del organigrama.
Resultado:		

Una vez ejecutados los casos de prueba, se llegó a la conclusión que todos los caminos básicos identificados fueron probados satisfactoriamente, demostrando que no existe código innecesario.

3.2.2 Pruebas Funcionales (caja negra)

Para la realización de estas pruebas se diseñaron casos de pruebas basados en requisitos, a partir de la técnica de partición de equivalencia definida en el Capítulo 1. Las pruebas de calidad fueron realizadas por el grupo de calidad del centro CEGEL perteneciente a la Facultad 3.

Se realizó una primera iteración donde se aplicaron los casos de prueba propuestos, detectándose por parte del grupo de revisores 32 no conformidades las cuales fueron corregidas. En una segunda iteración no se detectaron no conformidades, generándose de este encuentro el acta de liberación por parte de los revisores.

También se le mostró al cliente el sistema finalizado para que este evaluara el funcionamiento del mismo a través de la interacción con las distintas funcionalidades siendo el sistema propuesto de una alta aceptación para el cliente. De este encuentro se generó el acta de aceptación por parte del cliente.

A continuación se muestra un ejemplo de caso de prueba aplicados a los requisitos Autenticar Usuario, Eliminar Queja y Añadir Estudiante.

Tabla 16: Descripción del caso de prueba del requisito Autenticar Usuario.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*Usuario*	*Campo de Texto*	*No*	*Se escribe el nombre del usuario que se quiere registrar en el sistema. Solo se aceptarán letras y todas deben estar escritas en minúscula.*
2	*Contraseña*	*Campo de Texto*	*No*	*Se escribe la contraseña del usuario que se quiere autenticar en el sistema. Solo se aceptarán letras(mayúsculas y minúsculas),caracteres especiales y números*

Escenario	Descripción	Variable 1	Variable 2	Respuesta del sistema	Flujo central
EC 1.1 Registrar usuario y contraseña en el sistema	*Registrar satisfactoriamente el usuario en el sistema*	*V* *carlos*	*V* *ArturoelLoko123**	*Se registra satisfactoriamente el usuario y la contraseña*	*Selecciona la opción aceptar*
EC 1.2 Los datos son incorrectos	*Introduce datos incorrectos*	*I* *233**fbf*	*V* *asd4852*	*Identifica que los datos son incorrectos y no le da acceso*	*Selecciona la opción aceptar*
EC 1.3 Existen campos obligatorios vacíos	*Introduce datos incorrectos*	*V* *carlos*	*V*	*Indica que existen campos obligatorios vacíos*	*Selecciona la opción aceptar*
EC 1.4 la operación es cancelada	*La operación es cancelada*	*N/A*	*N/A*	*Cierra la interfaz*	*Selecciona la opción cancelar*

Tabla 17: Descripción del caso de prueba del requisito Añadir Estudiante

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*usuario*	*Campo de Texto*	*No*	*Se introduce el usuario del estudiante que se quiere añadir*

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Añadir el estudiante	*Se añade satisfactoriamente un estudiante*	*V* *cjperez*	*Se añade satisfactoriamente el estudiante*	*1-Selecciona la opción añadir estudiante 2-Selecciona la opción aceptar*
EC 1.2 Los datos son incorrectos	*El usuario introducido es incorrecto*	*I* *cj65*a*	*Identifica que el usuario es incorrecto*	*Selecciona la opción aceptar*
EC 1.3 El estudiante ya pertenece a un grupo	*El usuario introducido ya forma parte de un grupo*	*I* *rjrielo*	*Identifica que el estudiante ya forma parte de un grupo*	*Selecciona la opción cancelar*
EC 1.4 la operación es cancelada	*La operación es cancelada*	*N/A*	*Cierra la interfaz*	*Selecciona la opción cancelar*

Tabla 18: Descripción del caso de prueba del requisito Eliminar Queja

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*Descripción*	*Campo de Texto*	*No*	*Se escribe la descripción de la queja que se plantea*
2	*Estado*	*Campo de Texto*	*No*	*Se muestra el estado en el que se encuentra la queja(nueva, pendiente, resuelta)*

Escenario	Descripción	Variable 1	Variable 2	Respuesta del sistema	Flujo central
EC 1.1 Eliminar una queja del sistema	*Elimina satisfactoriamente una queja del sistema*	*V* *carlos plantea una queja*	*V* *nueva*	*Se elimina satisfactoriamente la queja del sistema*	*1- Selecciona la opción eliminar 2-Selecciona la opción aceptar*
EC 1.2 Eliminar una queja del sistema donde el estado sea pendiente o resuelta	*La queja ya a sido asignada a un área para darle seguimiento*	*V* *carlos plantea una queja*	*V* *pendiente*	*Identifica que la queja tiene un estado en el que no puede ser eliminada*	*Selecciona la opción aceptar*

EC 1.3 la operación es cancelada	*La operación es cancelada*	*N/A*	*N/A*	*Cierra la interfaz*	*Selecciona la opción cancelar*

3.2.3 Interfaces de la Aplicación

En las siguientes imágenes se muestran algunas interfaces de la aplicación.

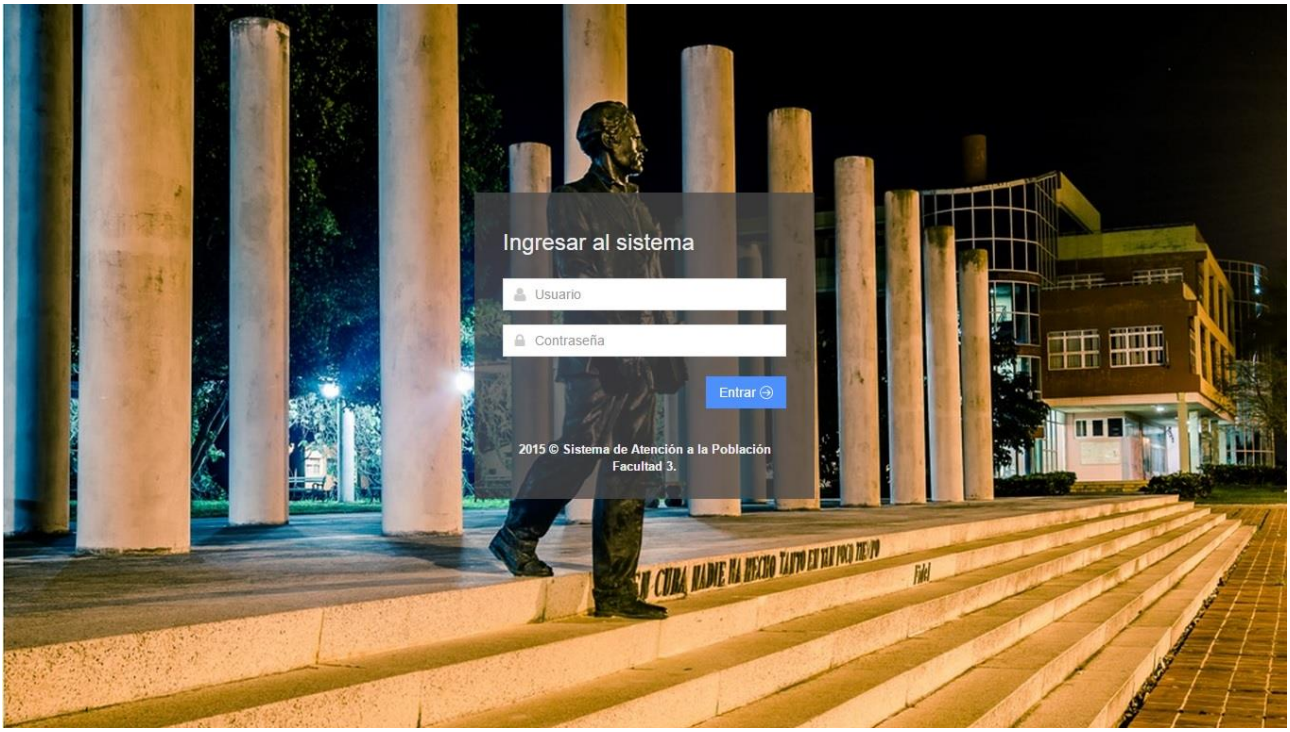


Ilustración 26: Interfaz perteneciente al login del sistema.

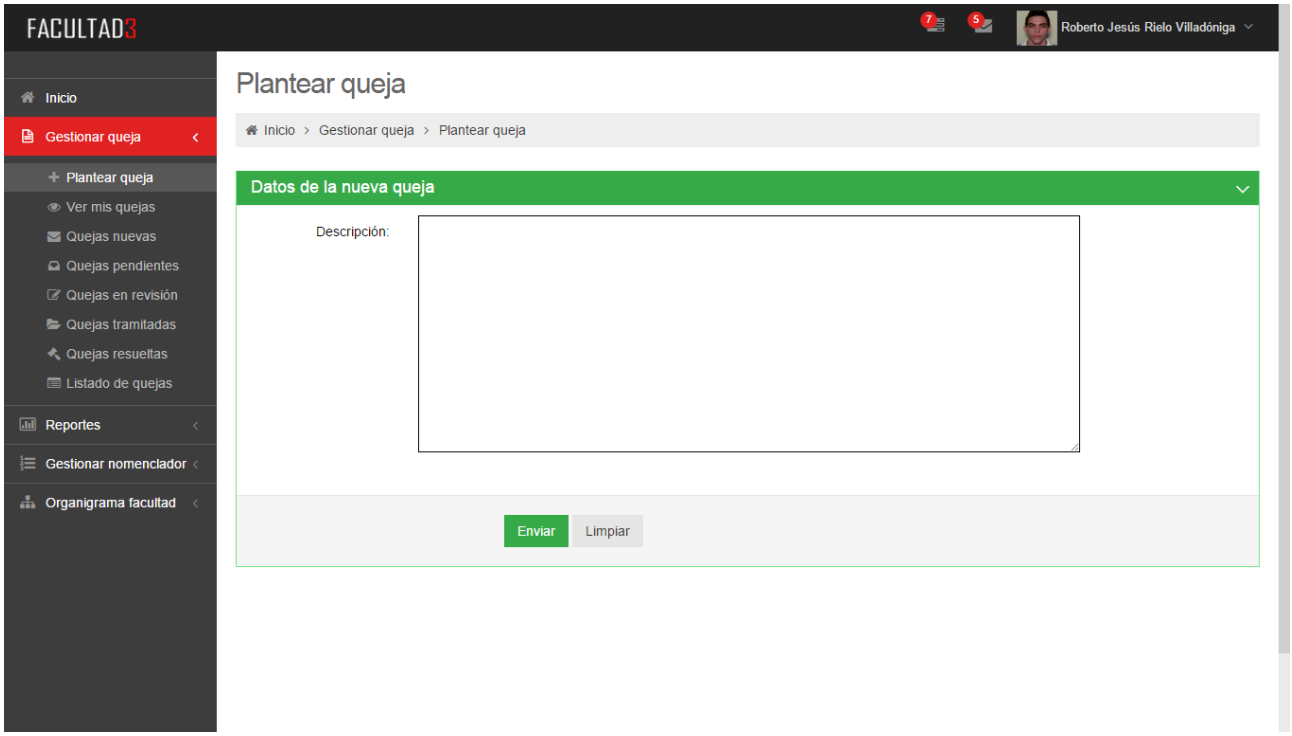


Ilustración 27: Interfaz perteneciente al planteamiento de una queja.

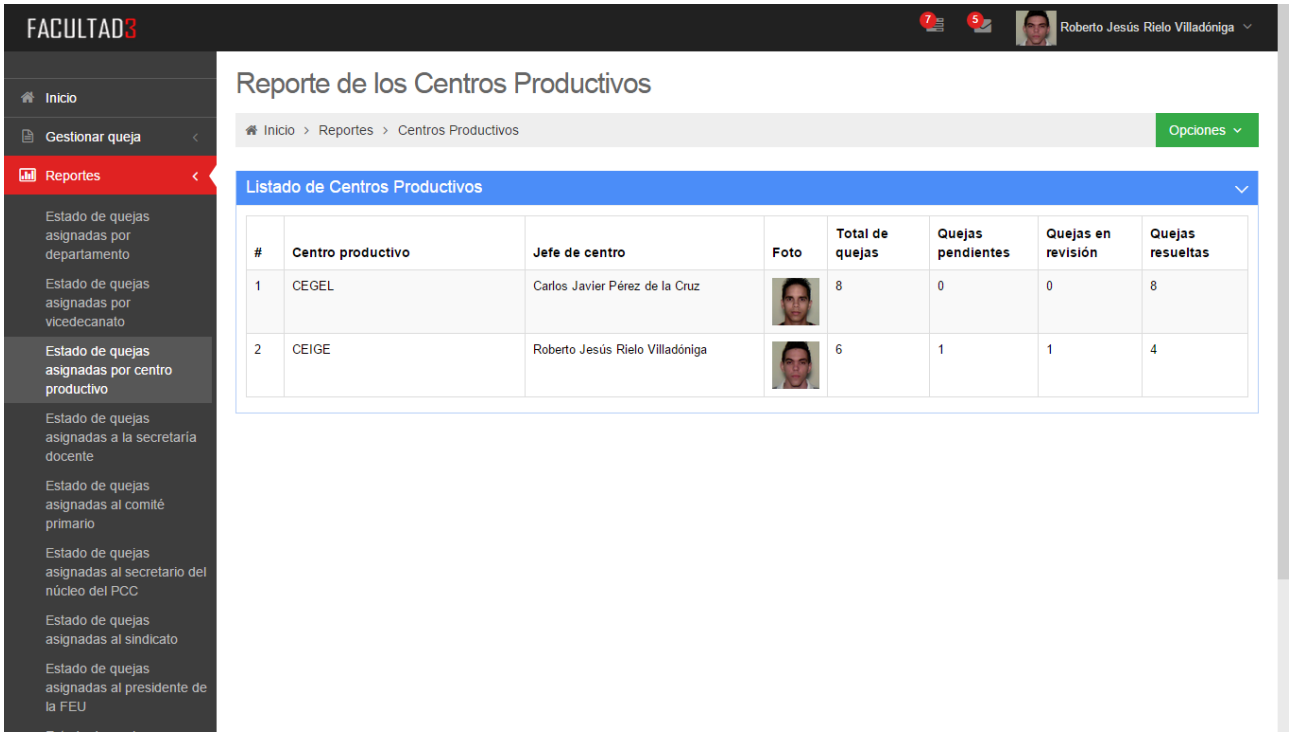


Ilustración 28: Interfaz perteneciente al reporte de los centros productivos.

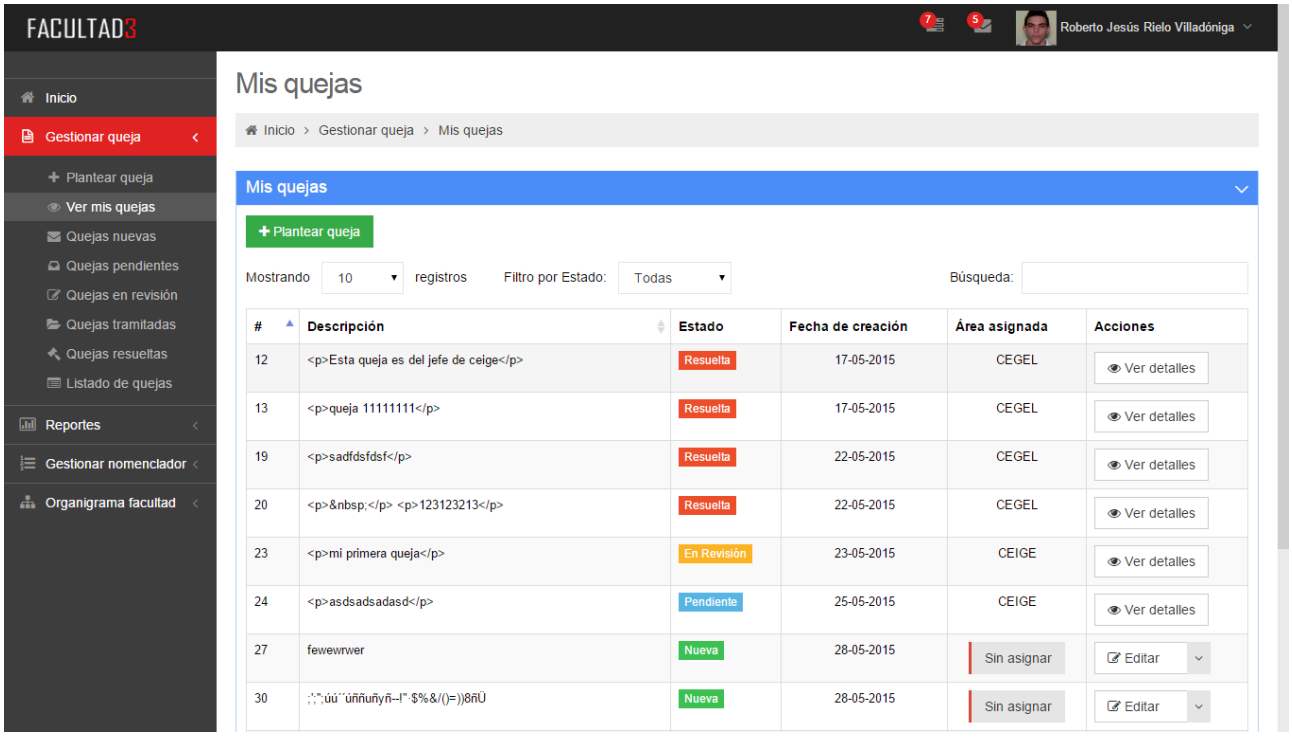


Ilustración 29: Interfaz perteneciente al listado de las quejas realizadas por el usuario.

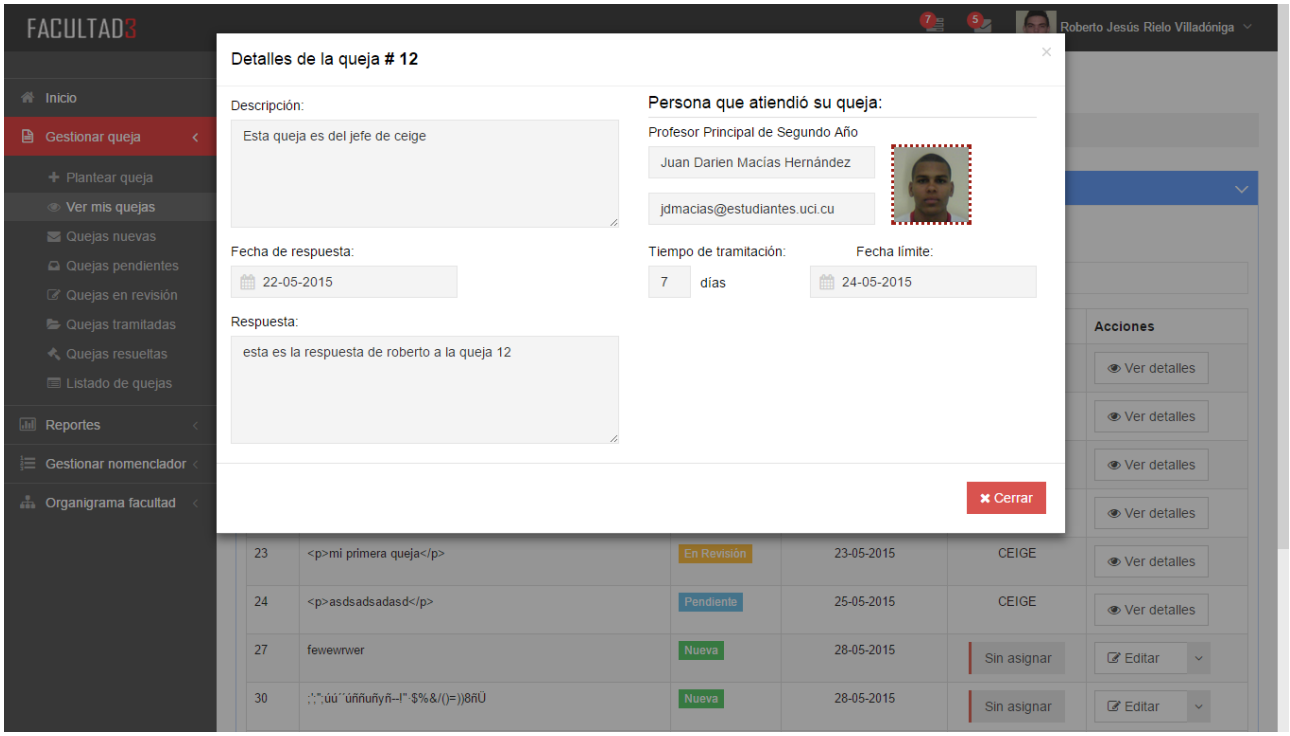


Ilustración 30: Modal de los detalles de una queja resuelta.

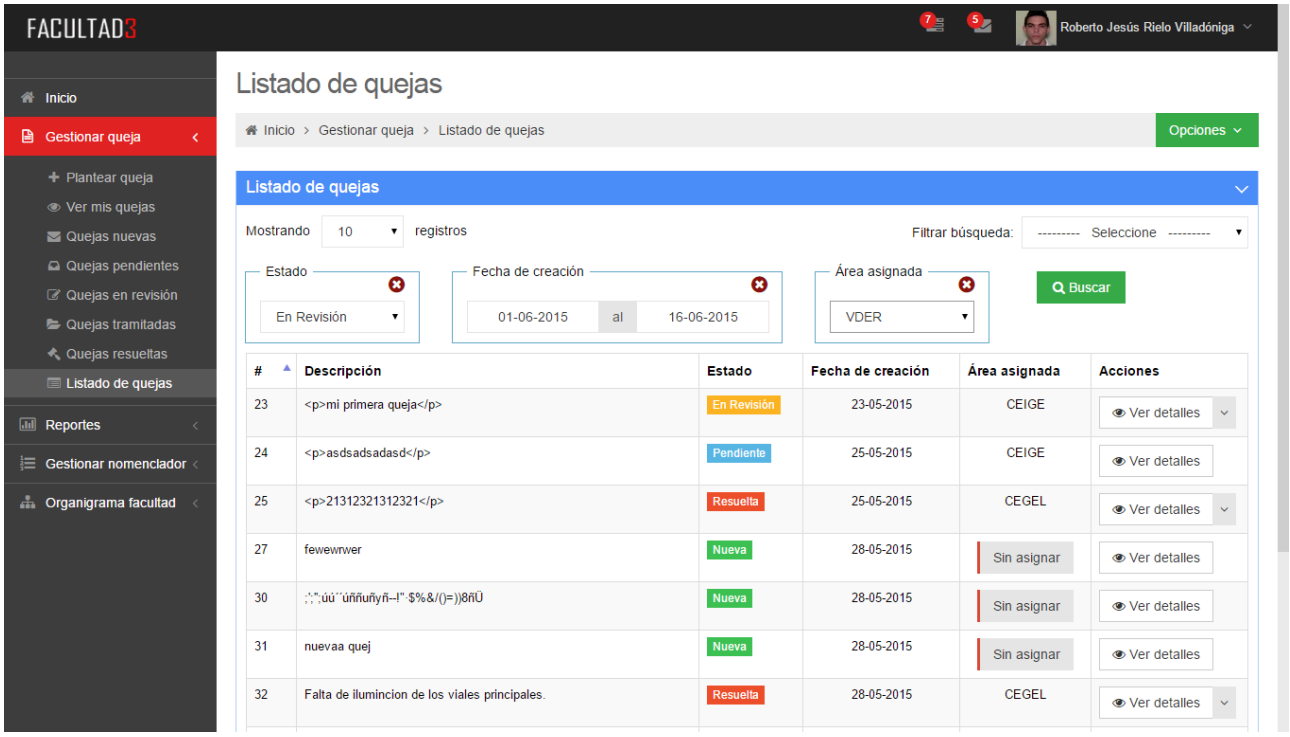


Ilustración 31: Interfaz perteneciente al listado de todas las quejas del sistema y las opciones de filtrado.

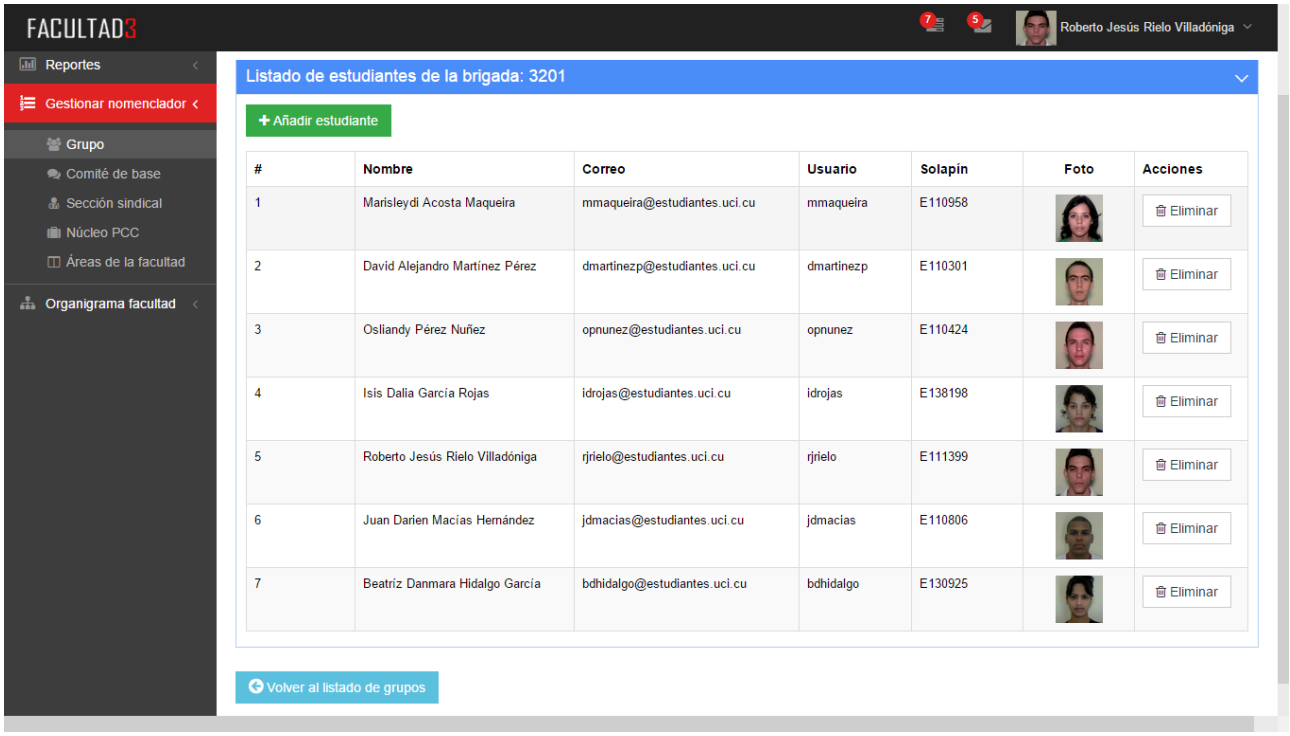


Ilustración 32: Interfaz perteneciente a la gestión de estudiantes por grupo.

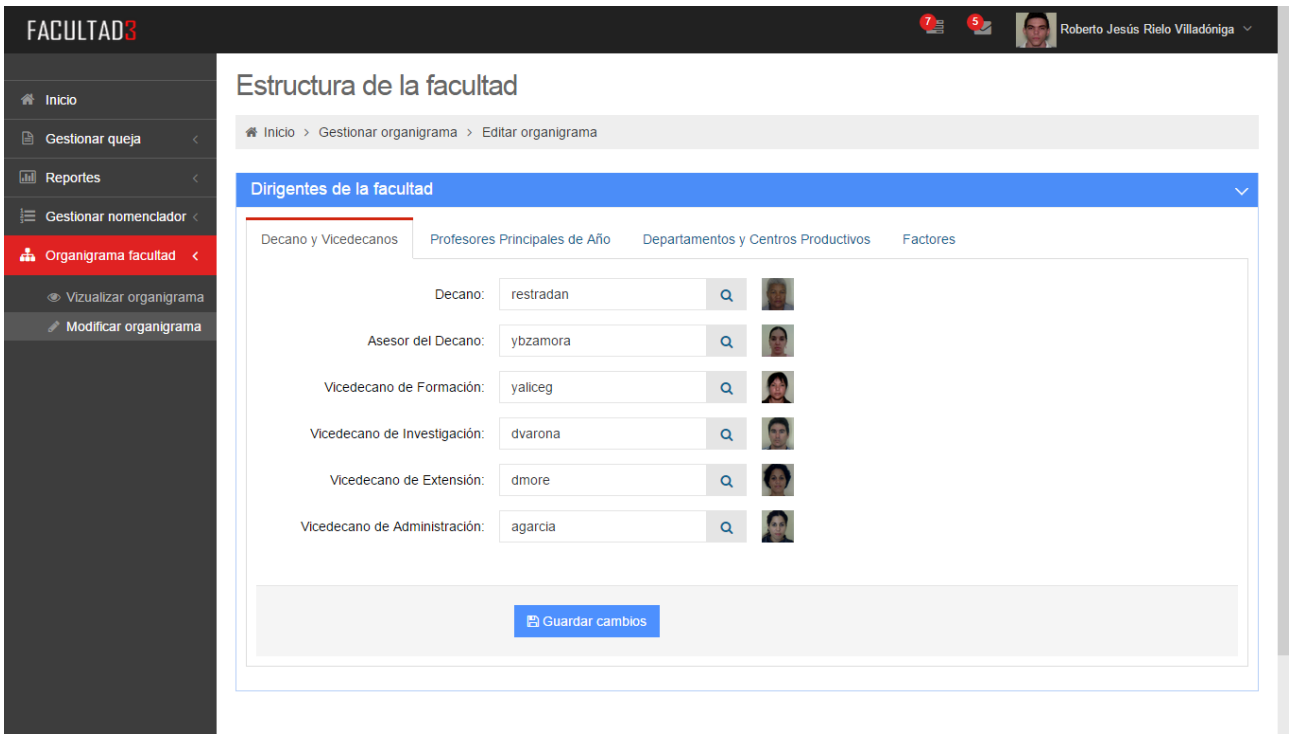


Ilustración 33: Interfaz perteneciente a la modificación del organigrama de la facultad.

3.2.4 Valoración de las Variables Dependientes

Actualmente la gestión de la información del proceso de atención a la población en la Facultad 3 se realiza de forma manual, no teniéndose un control exacto de la cantidad de quejas que se presentan, sus respuestas y los implicados en la misma, resultando limitado el acceso a la información generada y afectándose el seguimiento y control del proceso.

Para darle solución a estos problemas el sistema propuesto permite:

- Creación de quejas en la aplicación. El usuario puede además visualizar las quejas que ha generado, el estado en que se encuentra, las respuestas que se le han dado y los datos de la persona responsable de la respuesta.
- Define los estados por los que transita la queja (nueva, en revisión, pendiente, resuelta), permitiendo tener un control a los involucrados sobre los periodos por los cuales transita la queja y en qué estado se encuentra en un instante de tiempo.
- Cada queja es asignada a un área, la cual es la encargada de o dar la respuesta directamente o de tramitar la respuesta de varias áreas a la queja integrándolas en una sola respuesta.
- A través de la información que genera el sistema y mediante la incorporación de filtros para los datos guardados de cada queja, el sistema genera reportes que van desde una vista global del proceso hasta vistas especificas creadas por los usuarios.

Estas funcionalidades permitirán una mejora en el seguimiento y control del proceso en la Facultad 3.

CONCLUSIONES DEL CAPÍTULO

Luego de validar el diseño propuesto y la aplicación de las diferentes pruebas al software se puede llegar a la siguiente conclusión:

- Las técnicas de validación de requisitos empleadas permitió asegurar la fiabilidad de los estos.
- La aplicación de las técnicas TOC y RC brindaron resultados positivos, obteniéndose un diseño del sistema con baja complejidad de mantenimiento, un bajo acoplamiento entre sus clases y una alta reutilización.
- Se realizaron pruebas de caja negra y caja blanca que evaluaron y comprobaron la operatividad de las diferentes funcionalidades.
- La solución propuesta mejora el proceso de Atención a la Población en la Facultad 3 brindado un mayor control y seguimiento de la información generada en este proceso.

CONCLUSIONES GENERALES

La realización de este trabajo responde a la necesidad de buscar una solución al problema planteado. Terminada la investigación se llegó a las siguientes conclusiones:

- A través del estado del arte se evidenció que las herramientas existentes no se ajustan al entorno del problema de la investigación planteado, fundamentándose la necesidad de desarrollar un sistema para la gestión del proceso de atención a la población en la Facultad 3.
- El análisis de los elementos teóricos y técnicos que sustentaron la investigación permitió definir las herramientas, tecnologías y metodología a utilizar para el desarrollo del sistema.
- A partir de la metodología utilizada se generaron artefactos que permitieron guiar el desarrollo del sistema, utilizándose métricas para la validación de los mismos, obteniéndose resultados satisfactorios tanto para los requisitos como para el diseño.
- Las pruebas de software ejecutadas permitieron verificar el correcto funcionamiento de la aplicación, validándose que cumple con las especificaciones y requisitos definidos por el cliente.
- Se desarrolló el Sistema Informático para la Gestión de Información de la Atención a la Población en la Facultad 3 mejorándose el seguimiento y control de este proceso en la facultad.

RECOMENDACIONES

- Extender el uso de la herramienta al resto de las facultades de la universidad.
- Se le añada en futuras versiones la gestión de citas con los responsables de darle respuesta a las quejas.
- Se implemente la exportación de servicios como: listados actualizados de estudiantes y trabajadores por áreas.

Bibliografía

1. *Constitución de la República de Cuba.*
2. Salina Pareja, Maria Petronila. *Tesis a opción a MSc en Ciencias de la Educación.* 2008.
3. Zayas, Dr. Cs. Carlos Alvarez de. *METODOLOGIA DE LA INVESTIGACION CIENTIFICA.* 1995.
4. Ecured. [En línea] [Citado el: 2 de Noviembre de 2014.] http://www.ecured.cu.
5. Pérez, Annaliet Parra. *Ficha técnia del proyecto Fiscalía.*
6. Tribunal Supremo Popular. [En línea] [Citado el: 6 de Febrero de 2015.] http://www.tsp.cu.
7. Ministerio de Comunicaciones. [En línea] [Citado el: 6 de Febrero de 2015.] http://www.mincom.gob.cu.
8. Portal del Banco de la Nación del Perú. [En línea] [Citado el: 6 de Febrero de 2015.] http://www.bn.com.pe.
9. Mann, Mik. *Ingeniería del software.*
10. Beck, Kent. *Extreme Programming Explained.* 1999.
11. Mike Cohn, Addison Wesley. *User Stories Applied.* 2004.
12. Grady Booch, James Rumbaugh ,Ivar Jacobson. *El lenguaje unificado de modelado.* 2000. 8478290281.
13. Martin Fowler, Kendall Sccott,. *UML Gota a Gota.* 1999.
14. Sommerville, Ian. *Ingeniería del software.* 2005. Vol. Séptima Edición.
15. Jherson, Dickey. blogspot. *blogspot.* [En línea] 5 de 2013. [Citado el: 16 de 11 de 2014.] http://herramientascasecomparaciones.blogspot.com/.
16. Mardones, Paula y Panozo, Cecilia . slideshare. *slideshare.* [En línea] 8 de 4 de 2013. [Citado el: 16 de 11 de 2014.] http://es.slideshare.net/panozo1/herramientas-case-18430672.
17. Qué es un entorno de desarrollo integrado, IDE. [En línea] [Citado el: 6 de Febrero de 2015.] http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/.
18. [En línea] [Citado el: 15 de Enero de 2015.] https://netbeans.org.
19. Fabien Potencier, Francois Zaninotto. *Symfony, la guía definitiva.* 2008.
20. Eguiluz, Javier. *Desarrollo Web Águil con Symfony 2.* 2012.
21. Murphey, Rebecca. *Fundamentos de jQuery.* 2012.
22. Mark Otto, Jacob Thornton. *Bootstrap 3, El manual oficial.* 2014.
23. Pérez, Javier Eguíluz. *Introducción a XHTML.* 2008.
24. —. *Introducción a JavaScript.* 2008.
25. Eguiluz, Javier. *Introducción a CSS.*
26. [En línea] [Citado el: 20 de 11 de 2014.] http://www.maestrosdelweb.com/los-diferentes-lenguajes-de-programacion-para-la-web/.
27. Pacheco, Nacho. *Manual de Twig.* 2011.
28. [En línea] [Citado el: 20 de Nobiembre de 2014.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.
29. www.postgresql.org.es. [En línea] [Citado el: 20 de Noviembre de 2014.] http://www.postgresql.org.es/.
30. Team, Doctrine Project. *Doctrine 2 ORM Documentation.* 2011.
31. Ben Laurie, Peter Laurie. *Apache: The Definitive Guide.* III. 2002.
32. Foundation, The Apache Server. Documentación del Servidor HTTP Apache. [En línea] 2013. http://httpd.apache.org/docs/2.0/.
33. Control de Versiones. [En línea] [Citado el: 8 de Febrero de 2015.] http://producingoss.com/es/vc.html.
34. Source, El mundo es Open. El mundo es Open Source. [En línea] 2010. http://blogs.antartec.com/opensource/2008/11/antartec-es-open-source-parte-5-subversion-svn/.
35. Carlos Reynoso, Nicolás Kicillof. *Estilos y Patrones en la Estrategia de Arquitectura.* 2004.
36. Pressman, Roger S. *Software Engineering, a practitioner's approach.* VII. 2010.

37. Glenn E. Krasner, Stephen T. Pope. *Applications Programming in Smalltalk-80(TM):How to use Model-View-Controller (MVC).*
38. *Web-Application Development Using the Model/View/Controller Design Pattern.* Avraham Leff, James T. Rayfield.
39. Larman, Craig. *UML y Patrones.* 1999.
40. Pressman, Roger S. *Ingeniería del Software. Un Enfoque Práctico.* Sexta Edición. 2005. Vol. 5ta Edición.
41. Pfleeger, Charles P. *Security in computing.* 2006.
42. Estayno, M., Dapozo, G. y Cuenca Pletch, L. *Calidad de software e Ingeniería de Usabilidad.* 2012.
43. Campos, Mauricio. Extreme Programming. [En línea] Julio de 2004. [Citado el: 15 de Marzo de 2015.] http://apit.wdfiles.com/local--files/start/02_apit_xp_conceptos.pdf.
44. Gamma, Erich, y otros. *Design Patterns: Elements of Reusable Object-Oriented Software.* 1995.
45. Blaha, Michael. *Patterns of Data Modeling.* 2010.
46. Dittrich, K. *Conferencia seminario sobre base de datos orientadas.* 1994.
47. Fernández, R.J. *Modelo de Datos.*
48. Estándares de codificación en PHP - Aprende a Programar - Codejobs. [En línea] [Citado el: 11 de Marzo de 2015.] http://www.codejobs.biz/es/blog/2013/02/19/estandares-de-codificacion-en-php-psr0-psr1-psr2-y-psr3#sthash.11szU4rf.dpbs.
49. Sparx Systems - Tutorial UML 2 - Diagrama de Despliegue. [En línea] [Citado el: 15 de Marzo de 2015.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.
50. Ivar Jacobson, Grady Booch, James Rumbaugh. *El Proceso Unificado de Desarrollo de Software.* 2000.
51. Diagramas de componentes de UML: Referencia. [En línea] [Citado el: 15 de Marzo de 2015.] https://msdn.microsoft.com/es-es/library/dd409390.aspx.
52. Sommerville, Ian. *Ingeniería del software.* Séptima Edición. 2005.
53. Larman, Craig. *UML y Patrones.* México : s.n., 1999.
54. *Patrones Grasp. 2011; Available from: http://www.buenastareas.com/ensayos/Patrones-Grasp/1896730.html.*