Universidad de las Ciencias Informáticas
Vertex, Entornos Interactivos 3D, Facultad 4

Módulo Notificación y Monitoreo de usuarios para la plataforma Medicando

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

Autor: Daylilis Cardoso Roque

Tutores: Ing. Reinaldo García Maturell

Ing. Juan Gabriel Valdés DíazIng

Ing. Julio Cesar Espronceda Pérez

La Habana, 2018

Caminando en línea recta no puede uno llegar muy lejos.
Antonie de Saint-Exupéry

Dedicatoria

A mis padres por ser todo lo que quiero ser en mi vida, por merecer esta alegría tanto o más que yo, por tratarme siempre como su niñita linda, por todos los dolores de cabeza que les he dado y a pesar de eso encontrar siempre la manera de sacarme una sonrisa.

A mi sobrinito Dylan por ser mi hijo pequeño, por regalarme esa sonrisa siempre sincera.

Agradecimientos

A mis padres por ser todo en mi vida, por hacerme la persona que soy y por apoyarme en todo momento, por darme la familia linda que tengo y por nunca impedirme nada, por tratar de ponerse en mi lugar siempre. Pero por sobre todas las cosas, esperar siempre lo mejor de mí e impulsarme a conseguirlo.

A mi sobrinito Dylan por ser mi alegría constante y mi rayito de luz después de cada tormenta, por regalarme sus mejores momentos y por dejarme sentirme la tía más afortunada.

A mi familia en general por todo su apoyo, en especial a mi hermanito Daynier, mi abuelita Mimi y mi tía Sorangel, por apoyarme siempre, por su constante preocupación y por dejarme ser como soy sin pretender cambiarme.

A Leodán Luis Novoa por ser mi compañero incondicional en todos estos años. Por todo el amor brindado, por tanta paciencia en mis momentos de niñerías, por ser siempre la mano que me sustenta, por no rendirse nunca conmigo y por apoyarme en todo momento, no importa cuán difícil sea.

A mi familia en ciego por dejarme formar parte de ella todos estos años, por hacerme sentir como un miembro más y por su constante preocupación y apoyo.

A mis tutores por todas las noches de insomnio, por confiar en mí y hasta por secar mis lágrimas cuando los nervios me superaban, en especial a Juan Gabriel por tolerarme siempre con una sonrisa en la cara y no rendirse jamás.

A los hermanos que me ha regalado la vida, mis inseparables Nieve, Alejandro y Álvaro por aguantarme siempre, por tratar de entenderme por secar mis lágrimas y hasta por peliarme tanto, pero por sobre todas las cosas convertirme en una mejor persona.

A todos mis compañeros de aula (5101) por todas las experiencias vividas, por todos los momentos lindos que pasamos juntos, por estos cinco años de compañía, por ser mi familia.

A mis MLT por todo lo que me enseñaron, por tratar de entenderme, aunque algunos no lo lograron, por tratar de sacarme siempre una sonrisa.

En especial a mi morrito Sergio por ponerme todos los días las canciones que necesito oír, por entenderme y por ser sincero, aunque doliera.

A todos los profesores que contribuyeron en mi formación a lo largo de la carrera, por toda la preocupación y por soportar lo rápido q hablo y que nunca me cayo, por obligarme a dar lo mejor de mí y por su ejemplo cada día. A nuestra madre en esta escuela por preocuparse por cada uno de nosotros como si fuéramos sus propios hijos, por acompañarnos en cada momento, por defendernos como si fuera una gallina sacada, por estar siempre, por sufrir y reírse con nosotros, a nuestra profe Zaida. A el profesor que más me quiere en esta escuela, con el cual puedo contar siempre y trata de enseñarme algo en cada momento, en cada conversación, el profe que siempre tiene la mejor respuesta y que no importa cuántos años hayan pasado desde que me dio clases, a Rubén Alcolea.

A mi FAMILIA de la FEU por acompañarme siempre, por todas las noches de trabajo, pero de una alegría inmensa, a mi mejor presidenta Taire por ser siempre un ejemplo, por enseñarme a comportarme en todos los lugares, por ser una amiga de las que ya no vienen y por entenderme o por lo menos tratar.

Declaración de autoría

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

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

Daylilis Cardoso Roque
Autor

Ing. Reinaldo García Maturell
Tutor

Ing. Juan Gabriel Valdés DíazIng
Tutor

Ing. Julio Cesar Espronceda Pérez
Tutor

Resumen

El Centro de Entornos Interactivos 3D (VERTEX) de la Universidad de las Ciencias Informáticas ha desarrollado videojuegos web cuyo principal objetivo es ayudar en la rehabilitación de pacientes. Estos videojuegos almacenan información sensible para que el especialista pueda seguir la evolución de los pacientes. Este Centro desarrolló, además, la plataforma web Medicando; una aplicación que permite a los médicos administrar la información que brindan los videojuegos para seguir el desarrollo de los pacientes. La plataforma se basa en un desarrollo modular que permite agregar nuevos módulos sin afectar el funcionamiento básico de la aplicación para administrar Pacientes, Doctores, Tratamientos, Videojuegos y Especialidades. Sin embargo, la retroalimentación paciente-especialista se ve comprometida por la ausencia de funcionalidades que permitan mantener activo el flujo de información entre estos roles. El objetivo del presente trabajo es desarrollar el módulo Notificación y Monitoreo de usuarios para la plataforma Medicando y de esta manera mitigar la situación. Para la construcción de este módulo se utilizó el *framework* Yii en su versión 2.0, como Sistema Gestor de Base de Datos (SGBD) *MySQL* y para guiar el proceso de desarrollo se seleccionó la metodología *Extreme Programing* (XP).

Palabras clave: Medicando, módulo, notificación, monitoreo, trazas.

Índice general

Índice de figuras

Índice de tablas

Lista de códigos fuentes

Introducción

En los últimos años Cuba ha estado inmersa en el desarrollo de software para informatizar la sociedad, en especial en el área de la medicina. Facilitando, además, la realización de muchos procesos que contribuyen a producir una mejoría en la atención de los pacientes. En aras de apoyar este proyecto el centro VERTEX(Ver Anexo A.1), perteneciente a la Universidad de Ciencias Informáticas (UCI), ha desarrollado un grupo de videojuegos terapéuticos para favorecer los procesos de rehabilitación. Ejemplo de ello son los videojuegos Montaña Rusa Dinámica, Aventura Anatómica, Danzo-Terapia y Meteorix. En estos juegos no es posible llevar el control de la evolución de los pacientes basado en las estadísticas que generan.

Para dar solución a este problema VERTEX desarrolló la plataforma Medicando, un sistema multiplataforma que permite capturar los datos almacenados en los videojuegos que en el se integren y generar un conjunto de estadísticas grupales e individuales teniendo en cuenta los usuarios y los videojuegos que se administran. Este producto cuenta con un grupo de funcionalidades como son: la administración de usuarios y el establecimiento de roles a estos, lo que permite definir control de acceso y niveles de seguridad. Además, ofrece facilidades para la gestión de videojuegos, especialidades y tratamientos con el fin de proporcionar al personal de salud un medio para llevar el control de la evolución de los pacientes en sus tratamientos con videojuegos.

Desde un diagnóstico inicial que se realizó a Medicando, se detectan ciertas deficiencias que necesitan ser atendidas, por lo que se define como **situación problemática**:

- La plataforma no permite mantener informado a los usuarios de la realización de algún evento que los afecte directamente, como pueden ser alguna acción dentro de la gestión de videojuegos, especialidades y tratamientos, por lo que se ve comprometida la retroalimentación especialista-paciente.
- No se generan registros del comportamiento de los usuarios en el sistema lo que compromete la integridad de los datos.

Teniendo en cuenta lo antes mencionado se define como **problema de la investigación.** ¿Cómo contribuir a la retroalimentación paciente-especialista a partir del envío de notificaciones a los usuarios y la generación de trazas en la plataforma Medicando? Como **objeto de estudio** se define: Gestión de notificaciones y monitoreo de usuarios en plataformas web.

Para dar solución al problema de investigación planteado se define el siguiente **objetivo**: Desarrollar un módulo para la plataforma Medicando que permita enviar notificaciones a los usuarios, así como monitorear la actividad de los mismos en el sistema. Se propone como **campo de acción**: la generación de notificaciones

y monitoreo de usuarios en plataformas de rehabilitación a distancia.

Para dar cumplimiento al objetivo planteado se definen las siguientes tareas de investigación:

- Elaboración del marco teórico de la investigación a través de un estudio de sistemas homólogos de plataformas web atendiendo a la gestión de notificaciones y trabajo con trazas.
- Selección y estudio de las herramientas y tecnologías necesarias para el trabajo con notificaciones y el monitoreo de usuario.
- Análisis, diseño e implementación de la solución propuesta.
- Realización de pruebas unitarias, pruebas de integración y pruebas de aceptación de la solución propuesta.

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

Métodos teóricos:

- **Analítico-sintético:** para analizar desde diferentes aristas los conceptos asociados al trabajo con notificaciones y monitoreo de usuario en plataformas web. Este procedimiento permite describir las características generales y establecer las principales formas de retroalimentación entre usuarios de la plataforma.
- **Histórico-Lógico:** este método fue utilizado para estudiar los sistemas homólogos relacionados con la generación y envío de notificaciones y generación de trazas, lo que permitió adquirir conocimiento sobre la forma en que se realizan estas tareas para la definición de los antecedentes de la investigación.

Métodos empíricos:

- **Observación científica:** se emplea con el objetivo de observar el funcionamiento de algunas plataformas web orientadas a la generación de notificaciones; así como mecanismos de monitoreo de usuario que se utilizan en estas. Este método proporciona la vía para realizar un registro visual de las características comunes en este tipo de sistemas e identificar las que puedan formar parte de la solución.
- **Análisis estático:** este método fue utilizado para realizar un examen de la estructura del módulo implementado. Su utilización permitió además la aplicación de pruebas unitarias y de aceptación al módulo para detectar y corregir errores.
- **Consultas de fuente de información:** se utiliza para la consulta de fuentes bibliográficas durante la investigación.

El presente trabajo de diploma, está compuesto por tres capítulos, que incluyen los procedimientos desarrollados en relación con el trabajo investigativo, así como la propuesta de solución y validación de la investigación.

- **Capítulo 1** - Fundamentación Teórica: En esta primera etapa se aborda el estado del arte del tema que se investiga. Se destaca un estudio bibliográfico detallado sobre los principales conceptos asociados a

los mecanismos de notificaciones y monitoreo de usuario que utilizan las plataformas virtuales. En el desarrollo de esta etapa, se realiza el análisis del objeto de investigación y se seleccionan las tecnologías que se utilizarán en el desarrollo de la solución propuesta con el objetivo de incorporar nuevas funcionalidades asociadas al trabajo con notificaciones y monitoreo de usuarios en la plataforma Medicando.

- **Capítulo 2** - Características y diseño del sistema: Se describe el procedimiento seguido en las etapas de planificación y diseño que propone la metodología eXtreme Programming (XP) . Se especifican Historia de Usuario (HU), el plan de iteraciones y el plan de entregas, así como las tarjetas Contenido, Responsabilidad y Colaboración (CRC). También se describe la propuesta de solución y se mencionan los patrones de diseño y arquitectónicos utilizados en la implementación del módulo.
- **Capítulo 3** - Pruebas al Sistema: En esta etapa se definen las tareas de ingenierías correspondientes a cada HU. También se aplican las pruebas unitarias, pruebas de integración y pruebas de aceptación que permiten comprobar el correcto desarrollo de las funcionalidades implementadas.

CAPÍTULO 1

Fundamentación Teórica

En este capítulo se sintetiza la búsqueda y análisis de la información relacionada con el dominio del problema. Son descritos los conceptos fundamentales relacionados con la investigación y analizadas algunas soluciones existentes referentes al mismo entorno. En el desarrollo del mismo se seleccionan las tecnologías que se utilizarán en el desarrollo de la solución propuesta.

1.1. Medicando

Medicando es una plataforma web desarrollada con el objetivo de facilitar el seguimiento de pacientes en los tratamientos que les son asignados en el sistema, donde cada tratamiento debe estar relacionado con un videojuego de rehabilitación (Ver Anexo A.2). Medicando se basa en un mecanismo modular que permite incorporar subsistemas o módulos, sin afectar el funcionamiento del mismo. Esta herramienta fue desarrollada con tecnología PHP, para ello se utilizó como marco de trabajo el *framework* Yii en su versión 2.0; mientras que la gestión de los datos es administrada utilizando el Sistema de Gestión de Bases de Datos relacional (MySQL) y como servidor Apache Server [1].

Los roles que se definen en la plataforma son Paciente, Doctor y Administrador; donde cada uno puede ejecutar un grupo de operaciones específicas de acuerdo con los permisos que tiene asignado [2].

1.2. Notificación

La Real Academia Española define:

- **Notificar**: Dar noticia de algo o hacerlo saber con propósito cierto. Comunicar formalmente a su destinatario una resolución administrativa o judicial. Hacer a alguien destinatario de una notificación [3].
- **Notificación**: Acción y efecto de notificar. Documento en que consta la resolución comunicada [3].

En el ámbito de la informática, el proceso de notificación consiste en la acción de comunicar formalmente una resolución o noticia con propósito cierto, advirtiendo sobre determinadas situaciones a los usuarios del sistema. Su propósito fundamental es alertar que un evento en particular ha ocurrido. Una notificación se genera cuando el estado compartido de una aplicación cambia, es decir, si un usuario pone en conocimiento a los demás usuarios de un evento específico o tema en común [4]. Cuando se trabaja con notificaciones se debe considerar [5]:

- La forma que se provee el mecanismo de percepción y sincronismo.
- Además hay dos situaciones en las que se encuentran los usuarios: en el estado de notificadores o en el estado de observadores. Un usuario es notificador cuando pone en conocimiento a los demás usuarios de un evento. Un usuario es observador cuando recibe la notificación de un evento por parte de un usuario notificador.

Teniendo en cuenta lo anterior se puede concluir que los sistemas de notificaciones son soluciones que permiten el envío de mensajes o notificaciones a partir de suscripciones realizadas sobre algún evento específico.

1.3. Monitoreo de usuarios

El monitoreo de usuarios es la técnica para identificar y resolver los problemas de rendimiento de aplicaciones mediante la captura y análisis de la interacción con el navegador que tiene cada usuario en un sitio web. Este monitoreo provee una visión estratégica del comportamiento de una aplicación en el navegador del usuario final. El monitoreo de usuarios permite [6]:

- Ver la interacción de los visitantes en sitios web determinados, incluyendo las siguientes métricas: vistas únicas de páginas, tiempos de carga precisos y preferencias de navegador del usuario.
- Identificar y resolver problemas de rendimiento relacionados con usuarios específicos.
- Recopilar datos críticos y presentarlos de forma clara, precisa y significativa.
- Identificar errores en el navegador que normalmente son asintomáticos y podrían pasar desapercibidos.
- Capturar información detallada del código ejecutable en el navegador.
- Monitorear el impacto en el rendimiento de una página web que tiene la comunicación del navegador con servicios y APIs externos, como botones y *widgets* de redes sociales, mapas, o publicidad
- Comparar el comportamiento de una página web entre grupos de usuarios por geografía, sistema operativo, tipo de navegador y dispositivo.

En la plataforma Medicando el monitoreo de usuario se basará en grabaciones de la ocurrencia de eventos de importancia para el negocio mediante el uso de *Registros Log*.

1.3.1. Registros Log

En informática, se usa el término *Log Files*, historial o registro de *log*, a la grabación secuencial en un archivo o en una base de datos de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular (aplicación, *software*, actividad de una red informática, etc.). Estos registros *Log* constituyen una evidencia del comportamiento del sistema. Los mismos son presentados cronológicamente con datos adicionales que se utilizan para llevar estadísticas de uso de un determinado sitio, aplicación o *software* [7].

Tipos de registros log

- **De aplicaciones**: Los *logs* de aplicaciones graban cronológicamente las operaciones durante el funcionamiento de la aplicación. Su función forma parte de la lógica de la aplicación. Por lo tanto, no debería estar detenida durante el funcionamiento de la misma [7].
- **Del sistema**: Los registros de sistema graban cronológicamente los acontecimientos que sobrevienen a nivel de componentes del sistema [7].

Teniendo en cuenta las características descritas anteriormente de las notificaciones y el monitoreo de usuarios se realiza un breve estudio de algunos sistemas que las generan para comprobar si sus soluciones aplican al problema de la presente investigación.

1.4. Análisis de sistemas homólogos

En el presente epígrafe se realiza un estudio de sistemas homólogos basado fundamentalmente en el manejo y envío de notificaciones y el monitoreo de usuarios en plataformas web, con el objetivo de identificar los aspectos comunes y en los cuales se debe variar la solución propuesta.

Facebook

Es un servicio que permite conectar a los usuarios registrados en su página web, se pueden gestionar los espacios personales: crear álbumes de fotos, compartir videos, escribir notas, crear eventos o compartir el estado de ánimo con otros usuarios de la red. Además, tiene un componente importante de interactividad. Las notificaciones son actualizaciones sobre la actividad de *Facebook*. Los tipos de notificaciones que existen dependen de la plataforma o dispositivo que utilice el usuario para conectarse [8].

- **Ordenador y móvil**
  - *Notificaciones de alerta (de color rojo)*: Notificaciones que aparecen sobre los íconos con que cuente la aplicación. Cuando el usuario tiene una notificación nueva, aparecerá un pequeño ícono de color rojo con el número de las nuevas notificaciones que este ha recibido. Hay notificaciones independientes para las solicitudes de amistad y los mensajes, mientras que el resto de las notificaciones aparecerán sobre el ícono del globo [8].

- *Notificaciones de inserción*: Notificaciones que aparecen cuando no se está utilizando la aplicación de forma activa y ayudan a los usuarios a volver a interactuar con sus amigos [8].
- *Notificaciones de correo electrónico*: Notificaciones que se reciben por correo electrónico donde se avisa sobre actualizaciones o comentarios [4].

- **Solo en el ordenador**
  - *Notificaciones emergentes*: Notificaciones que aparecen en la pantalla cuando el usuario ha iniciado sesión en la aplicación y un amigo interactúa con el mismo, ya sea a partir de un mensaje directo o por interacción con el perfil del usuario autenticado [8].

Twitter

Es una red social en línea que permite a los usuarios enviar y leer mensajes cortos de 140 caracteres llamados *tweets*. Los usuarios registrados pueden leer y publicar *tweets*, pero los que no están registrados sólo pueden leerlos. Los usuarios acceden a *Twitter* a través de la interfaz web, SMS o aplicación para dispositivo móvil [9]. Actualmente este sistema cuenta con más de 350 millones de usuarios activos mensuales. *Twitter* gestiona diferentes tipos de notificaciones, dentro de estas destacan:

- **Notificaciones que se envían como mensajes de texto SMS**: Una vez que los usuarios tienen un número de teléfono asociado a la cuenta, pueden configurar en sus perfiles la opción de recibir notificaciones mediante mensajes de texto [9] [4].
- **Notificaciones web y de navegador**: Una notificación web es un mensaje emergente que aparece en el navegador web para notificar a los internautas sobre lo que sucede en esta red social [9].
- **Pestaña de notificación**: ***Pestaña Activity***, que es un sistema de notificaciones donde se ven *retweets*, *tuits* favoritos y nuevos seguidores, pero de aquellos usuarios a los que seguimos [4].

Sistema de Información Clínico-Hospitalaria del hospital Padua, Italia.

Posee un componente computarizado de notificaciones, envío de alertas y SMS, que comenzó a utilizarse en enero del 2008 y fue desarrollado por la Sociedad de Procesamiento de Datos Médicos (GMD, por sus siglas en alemán). Este sistema gestiona todas las peticiones realizadas por los médicos y reúne la información procedente de todos los servicios de diagnóstico, incluyendo el laboratorio clínico y los departamentos de imagen. En tiempo real, se notifican a los médicos, de forma automática, los resultados del laboratorio, mediante un mensaje de correo electrónico. Para el envío de dichos resultados, se generan además dos mensajes: un SMS hacia el teléfono móvil del médico de turno, y otro a nivel de departamento, que llega al monitor del clínico que ordenó el análisis. Ambos tipos de mensaje (el SMS y la alerta) incluyen los códigos apropiados para la identificación del paciente y especifican el número de teléfono móvil del médico de guardia en el laboratorio. El sistema de información de la institución lleva una constancia del éxito o el fracaso de cada proceso de notificación [10].

SALUS

Es un *software* para la gestión integral de clínicas, centros médicos y hospitales [1], que permite gestionar de forma global el conjunto de áreas de gestión de un centro de salud. Para facilitar la organización y comunicación entre profesionales del mismo centro, SALUS implementa un sistema de mensajería entre usuarios del sistema. Cada mensaje puede tratarse como una tarea a realizar, quien la envía siempre tiene confirmación de cuándo ha sido vista por el receptor y cuándo la marca como realizada. Permite además comunicar datos de manera automática a los pacientes, como por ejemplo, el recordatorio de una cita próxima o el resultado de una prueba diagnóstica, estos mensajes se envían directamente a su teléfono móvil [11].

1.4.1. Conclusiones del análisis de sistemas homólogos

Teniendo en cuenta los elementos comunes en las diferentes aplicaciones analizadas, se puede concluir que la plataforma Medicando precisa dos canales de comunicación con los usuarios, estos son:

- Envío de mensajes a los correos de cada usuario que se encuentre involucrado con determinada acción.
- Generación de alertas de sistema que permitan comunicar los cambios de estado del mismo.

Además se prevee ofrecer la posibilidad de configurar el recibo de notificaciones de los usuarios del sistema. En ninguna de las aplicaciones analizadas se pudo comprobar la generación de registro de las actividades de los usuarios, debido a que no se pudo acceder como administrador del sistema. No obstante se determina que la plataforma Medicando precisa un mecanismo para la generación de trazas mediante registros log de sistemas, los cuales grabarán en base de datos la ocurrencia de eventos que se declaran de importancia para el negocio, por lo que se puede emplear como una herramienta de análisis ante los fallos que puedan ocurrir.

1.5. Selección de tecnologías para el envío de notificaciones

Atendiendo a la necesidad de establecer un sistema de notificaciones en la plataforma Medicando, a continuación, se realiza un estudio de las tecnologías en correspondencia con el tipo de notificación dependiendo del canal de entrega:

A nivel de correo

El *framework* Yii provee una extensión propia que cuenta con las facilidades necesarias para realizar el envío de correos electrónicos basados en esquemas de vistas y patrones. Cuenta con una documentación abundante y un constante soporte que la hacen ideal para dar cumplimiento a parte del objetivo de la presente investigación [12] [13]. Teniendo en cuenta las características mencionadas se selecciona la extensión ***SwiftMailer*** para implementar el envío de notificaciones vía correo electrónico en la plataforma Medicando.

[1] Desarrollado por la empresa española QSOFT, Empresa desarrolladora de software dedicados a la atención médica

A nivel de sistema

Para la selección de la biblioteca javascript a utilizar, se analizan 5 bibliotecas atendiendo a su nivel de uso [14]. Se tienen en cuenta los siguientes parámetros para establecer la comparación:

- **Facilidad de uso**: indica el nivel de complejidad que requiere su utilización.
- **Opciones de posicionamiento**: indica el nivel de opciones de configuración para establecer el posicionamiento del mensaje en la web.
- **Efectos visuales**: indica el nivel de opciones de configuración de efectos visuales para mostrar el mensaje en la web.
- **Documentación**: indica el volumen de documentación existente para la utilización de la tecnología.

Tabla 1.1. Comparativas de tecnologías para el envío de notificaciones a nivel de sistema

Criterio/Tecnología	Alertify.js	ohSnap!	Toastr	iziToast	AmaranJS
Facilidad de uso	8	6	7	8	7
Opciones de posicionamiento	7	4	7	8	9
Efectos visuales	5	9	6	10	5
Documentación	4	6	9	10	8
Total	24	25	29	36	29

Luego de realizar el estudio comparativo queda seleccionada la biblioteca **iziToast** disponible desde GitHub para desarrollar la generación de alertas de sistema. Esta biblioteca resulta ser la que mejor se adapta al *framework* y mayor opciones de configuración presenta [15], aunque el mayor peso de la decisión es la abundante documentación disponible en IZITOAST y la potente comunidad de desarrollo que respalda la misma.

1.6. Herramientas y tecnologías a utilizar

Considerando el estudio realizado previo a la presente investigación, se asumen y adoptan las herramientas y tecnologías utilizadas por el equipo de desarrollo de la última versión de la plataforma Medicando. Las principales herramientas a utilizar para llevar a cabo el proceso de desarrollo de la solución se describen a continuación. [1].

- **Marco de trabajo**: Un framework proporciona estructura al código fuente, forzando al desarrollador a crear un código más legible y más fácil de mantener [16]. Se utilizó el *framework* Yii en su versión 2.0. Yii es un *framework* basado en componentes, escrito en PHP, para el desarrollo rápido a gran escala de aplicaciones web [17]. Permite la máxima reutilización en la programación web y acelera de manera significativa el proceso de desarrollo de aplicaciones [18]. Las aplicaciones se construyen bajo el patrón arquitectónico Modelo-Vista-Controlador (MVC). Sobresale en comparación con otros

frameworks PHP en ser eficiente, rico en características y en estar bien documentado [19]. Esta herramienta soporta como lenguajes de programación *hypertext markup language* (HTML), *Cascade Style Sheet* (CSS), PHP v5 y *JavaScript* [20].

- **Entorno de desarrollo**: Fue utilizado *NetBeans 8.2*, esta herramienta permite el desarrollo rápido y fácil de aplicaciones Java de escritorio, móviles y aplicaciones web, así como aplicaciones HTML con JavaScript y CSS. El IDE también proporciona un gran conjunto de herramientas para desarrolladores de PHP y C/C ++. Es gratuito y de código abierto, además es objeto de colaboración de una gran comunidad de usuarios y desarrolladores de todo el mundo [21].
- **Sistema Gestor de Base de Datos**: Los SGBD están constituidos por un paquete de software cuya función es la gestión del acceso a la base de datos (BD), las operaciones fundamentales son: crear, modificar, eliminar y obtener la estructura asociada al esquema lógico [22]. Se utilizó MySQL, este es un sistema de gestión de bases de datos relacional. Su diseño multihilo le permite soportar una gran carga de forma eficiente. Es uno de los gestores más usados en el mundo del *software* libre, debido a su rapidez y facilidad de uso. Esta gran aceptación, es debida a que existen infinidad de bibliotecas y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración [23]. Las características fundamentales que refleja para su elección por encima de otros gestores de base de datos son: el coste gratuito, la velocidad operacional, facilidad de uso y de integración con la mayor parte de los entornos de programación, la existencia de una nutrida y activa comunidad [24].

1.6.1. Servidor Web

Un servidor web es un programa que implementa el protocolo *hypertext transfer protocol* (HTTP). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

XAMPP

XAMPP es un paquete que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y *Perl*. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MariaDB, PHP, Perl. Desde la versión "5.6.15", XAMPP cambió la base de datos de MySQL a MariaDB. El cual es un *fork* de MySQL con licencia GPL [25].

1.7. Metodología de Desarrollo de Software

Las metodologías de desarrollo de software son enfoques de carácter estructurado y estratégico que permiten el desarrollo de programas con base a modelos de sistemas, reglas, sugerencias de diseño y guías [26]. Una metodología de desarrollo de software brinda al equipo de trabajo un marco para construir aplicaciones de manera eficiente y rigurosa, garantizando un producto cercano al esperado. Si no se desarrolla a partir de una metodología, el resultado final será impredecible y no se podrá controlar el avance del proyecto [27]. A continuación se evalúa la metodología de desarrollo XP que aplica en los procesos ingenieriles de desarrollo ágil.

XP

XP es una metodología ágil para el desarrollo de software y consiste en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo, centrada en potenciar las relaciones interpersonales como clave para el éxito del desarrollo de software [28]. La filosofía de XP es satisfacer al completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo.

Faculta a sus desarrolladores para responder con seguridad a las necesidades cambiantes de los clientes, incluso en etapas tardías del ciclo de vida del producto. Los jefes de proyecto, clientes y desarrolladores son socios iguales en un equipo de colaboración que se auto-organiza en torno al problema a resolver de la forma más eficiente posible. XP mejora un proyecto de software en cinco aspectos esenciales; la comunicación, la sencillez, la retroalimentación, el respeto y el valor [29].

La programación extrema se basa en trece prácticas básicas que deben seguirse al pie de la letra. Las cuales son: equipo completo, planificación, test del cliente, versiones pequeñas, diseño simple, pareja de programadores, desarrollo guiado por las pruebas automáticas, mejora del diseño, integración continua, el código es de todos, normas de codificación, metáforas, ritmo sostenible [30].

Se considera XP como la metodología idónea para llevar a cabo el proceso ingenieril, debido a que sus características aplican en el desarrollo de la solución propuesta como se demuestra a continuación. Cabe destacar que serán adoptadas la 4 fases propuestas (Planificación, Diseño, Desarrollo y Prueba) [31].

- **Tipo de Desarrollo Iterativo e incremental**: El método está basado en lo que son las mejoras continuas, a base de iteraciones y de un desarrollo incremental al estilo espiral permitiendo al equipo de desarrollo dividir el trabajo a realizar en cada una de las iteraciones.
- **Trabajo en Equipo**: Más específico todavía, es el trabajo en parejas, el objetivo es que el enfoque en parejas sea mayor. El equipo de desarrollo en este caso es solo conformado por un desarrollador, pero cumple con ser un equipo de desarrollo pequeño que algo que señala esta característica.
- **Alguien del equipo trabaja con el cliente**: Es fundamental que el cliente intervenga en el desarrollo, pero obviamente él no estará en la sala de desarrollo, se debe asignar a una persona que sea la encargada de tener las reuniones con el cliente de forma constante. Este será quien comunique al equipo los cambios o el seguimiento del proyecto. En el desarrollo de la propuesta de solución, desde el comienzo se realizan reuniones periódicas con el cliente, donde se expone cada avance en el desarrollo y donde se obtienen cada uno de los cambios sugeridos por el mismo.
- **Corrección de Errores**: El hecho de que la metodología XP sea rápida para el desarrollo, no significa que se pasen por alto los errores, de hecho, primero se le tiene que dar corrección a los errores antes de seguir avanzando en el proyecto. Esta característica es aplicada cada vez que se detectan errores en el desarrollo del producto final, estos errores pueden ser detectados por el propio desarrollador o por el cliente en cada una de las reuniones que se desarrollen.
- **Reestructuración del Código**: El principal objetivo de la reestructuración es simplificar el código, pero no las funciones. Para el desarrollo del producto final está es una característica muy aplicada, debido a que solo es un desarrollador y el principal objetivo es simplificar el código.

1.8. Conclusiones del capítulo

A partir del estudio de diferentes aplicaciones web y la observación del comportamiento de los módulos de notificaciones, fue posible arribar a las siguientes conclusiones:

- Se identificó como principales vías para el envío de notificaciones, la generación de mensajes de alerta por el propio sistema y el despacho de correos electrónicos.
- Para solucionar las insuficiencias detectadas se selecciona *iziToast* como biblioteca *javascript* que permitirá mostrar los mensajes en la plataforma, mientras que la extensión *SwiftMailer* facilitará el envío de correos electrónicos.
- Se adopta como marco de trabajo el *framework* Yii en su versión 2.0, como IDE de desarrollo *NetBeans* v8.2 . Además, como complemento para garantizar el correcto funcionamiento del *framework*, se selecciona el paquete de programas XAMPP, que incluye el *Apache Server* como servidor web, PHP como lenguaje de programación y MySQL como SGBD.
- Se acoge XP como metodología de desarrollo, ya que garantiza la generación de artefactos que son necesarios para una mayor comprensión de la solución a desarrollar.

CAPÍTULO 2

Características y diseño del sistema

En el presente capítulo se presenta una propuesta de solución para los problemas detectados en el estudio del estado del arte realizado en el capítulo anterior. También se definen las HU, la planificación de entrega de versiones del producto y el diseño del sistema de acuerdo a las fases que propone la metodología seleccionada.

2.1. Propuesta de solución

Luego de haber analizado las necesidades del sistema y seleccionado las herramientas para la implementación, se define el módulo a desarrollar para dar solución al problema planteado.

- **Módulo Notificación y Monitoreo de Usuarios:** responderá por la necesidad de informar a los usuarios del sistema, ante la ejecución de acciones en la aplicación, que afecten información sensible relacionada con estos. También mantendrá un control oportuno de las principales actividades que realizan los usuarios en la plataforma, permitiendo llevar un control sobre la integridad de los datos del sistema.

2.1.1. Principales funcionalidades

El módulo propuesto, de manera general, garantiza un grupo de funcionalidades para su correcto funcionamiento, estas son:

- *Generación de Notificaciones:* Este paquete funcional permite al sistema poder enviar notificaciones a cada uno de los usuarios registrados en el mismo una vez que ha sucedido un evento que se haya declarado de alta importancia en el negocio o que los afecte directamente. Permite además que los doctores, de forma directa puedan enviar notificaciones a sus pacientes, una vez que este crea necesario comunicarles recomendaciones sobre sus tratamientos.
- *Configuración de recibo de Notificaciones:* Este paquete funcional permite a cada usuario del sistema poder seleccionar el medio mediante el cual desea recibir las notificaciones, se precisan en la solución

propuesta dos canales de envió de notificaciones, mediante correo electrónico y mediante el propio sistema una vez que el usuario esté registrado y activo en el mismo.

- *Generación de Notificaciones vía correo electrónico:* Este paquete funcional permite generar notificaciones que solo usarán como canal de envío el correo electrónico, siempre teniendo en cuenta la configuración personal de cada usuario afectado por la notificación generada.
- *Generación de Notificaciones vía sistema:* Este paquete funcional permite generar notificaciones que solo usarán como canal de envío el propio sistema, ocurren además una vez que el usuario está registrado y activo en el mismo, se comprueba siempre que el usuario activo tenga configurado el recibo de este tipo de notificación.
- *Mostrar Notificaciones:* Este paquete funcional permite que cada usuario pueda ver las notificaciones pendientes que tiene; ofrece además la posibilidad de marcarlas todas como leídas una vez vistas, y acceder a todas sus notificaciones.
- *Listar Notificaciones:* Este paquete funcional permite mostrar en una lista todas las notificaciones que tiene un usuario, y brinda la opción de eliminarlas. Permite además mostrarle al administrador del sistema una lista de todas las notificaciones que se han generado sin importar los canales utilizados.
- *Generar Trazas:* Este paquete funcional permite al sistema poder generar trazas para cada uno de los sucesos que se declararen de alta importancia en el negocio.
- *Listar Trazas:* Este paquete funcional permite mostrar en una lista todas las trazas que se han generado. A esta lista solo tiene acceso el administrador del sistema.

Con el desarrollo del módulo propuesto se ofrece además de las funcionalidades descritas, un grupo de funciones de interfaces que permiten facilitar el trabajo con el mismo.

Funciones Interfaces

Para del envío de notificaciones, se ofrece una interfaz que permite a otros módulos integrados a la plataforma, hacer uso del sistema de envío de notificaciones en caso que lo requieran. La interfaz ***Notifications.php***, provee los principales métodos utilizados para la generación de notificaciones como se muestra en la figura **??**.

El método ***createNotification*** se encarga de crear y registrar una notificación en la base de datos de la plataforma. Recibe como parámetros el tipo de notificación que puede ser *success o warning* en dependencia de la acción realizada, el título de la notificación, el mensaje contenido dentro de la misma y un arreglo de usuarios a notificar. El método crea una notificación temporal en la que inicializa además de los atributos pasados por parámetros la fecha en la que se realiza la acción mediante una función nativa de PHP , el usuario origen de la notificación mediante una funcionalidad propia del *framework* y el estado de la notificación como *pendiente*. Posteriormente comprueba que el arreglo de usuarios está *seteado*, confirma que el valor sea un arreglo y crea para cada uno de los usuarios destino una nueva notificación a la que le pasa los atributos de la notificación temporal antes creada.

Código fuente 2.1. Método de generación de notificación

```
1  public  static  function  createNotification ( $type , $title , $message , $users =
2  $noty = newNotification();
3     $noty ->key = $type ;
4     $noty -> title = $title ;
5     $noty ->created_at = date ( "Y-m-d H: i : s " ) ;
6     $noty ->message = $message ;
7       $noty->user_from_id = Yii::$app->user->id;
8       $noty->readed = false;
9
10      if(isset($users) && is_array($users)){
11
12    foreach ($users as $to){
13    $temp = new Notification();
14    $temp->attributes = $noty->attributes;
15    $temp->user_to_id = $to;
16    $temp->save();
17    ....
18
```

Para la generación de trazas del sistema, igualmente se ofrece la interfaz (***Notifications.php***) que permite generar trazas en susbsistemas o módulos que se integren posteriormente. Por lo que la interfaz proporciona los principales métodos utilizados para la generación de trazas como se muestra en la figura **??**.

El método ***saveSystemLog*** es el encargado de crear y registrar una traza en la base de datos de la plataforma. Recibe como parámetros el modelo sobre el cual se realiza la acción y la acción realizada. El método crea una traza en la que inicializa además de los atributos pasados por parámetros la fecha en la que se realiza la acción mediante una función nativa de PHP y el usuario que la realiza mediante una funcionalidad propia del *framework*.

Código fuente 2.2. Método de generación de trazas

```
1  public function saveSystemLog($model_to,$action){
2
3      $log = new SystemLogs();
4      $log->date_time = date('Y-m-d H:i:s');
5      $log->user_id = Yii::$app->user->identity->getId();
6
7     $log->action = "Se ha " . $action . " el elemento
8      (".$model_to->name .") del modelo ".
9      $model_to->className() .".";
10    $log->save();
11    }
12
```

Para el correcto funcionamiento del módulo Notificación y Monitoreo de Usuarios las llamadas a los

métodos de generación de notificaciones y trazas, quedan definidas como muestra en la figura **??**

Código fuente 2.3. Algoritmo para hacer llamadas a los métodos de generación de notificaciones y trazas

```
1 if (($noty = Yii::$app->getModule('notifications')) !== null) {
2     $noty->createNotification(...);
3
4     $noty->saveSystemLog(...);
5     }
6
```

2.1.2. Principales eventos que generan notificaciones y trazas

Para una mejor comprensión del módulo Notificación y Monitoreo de Usuarios, se seleccionaron los principales eventos que por su importancia en el negocio deben generar notificaciones a cada uno de los usuarios correspondientes. Así como la generación de trazas de los mismos, seguidamente se muestran en la tabla:

Tabla 2.1. Eventos que generan notificaciones y trazas

No	Eventos que genera la notificación y trazas	Receptor de las notificaciones
1	Crear un videojuego	Doctor
2	Actualizar un videojuego	Doctor
3	Eliminar un videojuego	Doctor
4	Crear un tratamiento	Doctor
5	Actualizar un tratamiento	Doctor
6	Eliminar un tratamiento	Doctor
7	Asignar tratamiento a un paciente	Paciente,
8	Eliminar la asignación de un tratamiento a un paciente	Paciente,
9	Comenzar un tratamiento	Paciente, Doctor
10	Culminar un tratamiento	Paciente, Doctor
11	Crear una especialidad	Doctor
12	Actualizar una especialidad	Doctor
13	Eliminar una especialidad	Doctor

2.2. Especificación de requisitos de software

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Los requisitos reflejan la necesidad de los clientes de un sistema que ayuda a resolver problemas como el control de un dispositivo, hacer un pedido o encontrar información. El proceso que permite descubrir, analizar, documentar y verificar esos servicios y restricciones se denomina ingeniería de requisitos [32].

2.2.1. Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer [32].El levantamiento de requisitos para el módulo propuesto arrojó como requisitos funcionales los siguientes:

Tabla 2.2. Requisitos Funcionales

No	Requisito Funcional
RF1	Generar noticaciones
RF2	Listar notificaciones
RF3	Eliminar notificación
RF4	Mostrar notificaciones pendientes
RF5	Marcar notificaciones como leídas
RF6	Visualizar una notificación seleccionada
RF7	Eliminar notificaciones de manera simultánea
RF8	Configurar recibo de notificaciones
RF9	Enviar notificaciones a correo electrónico
RF10	Enviar notificaciones a pacientes
RF11	Generar trazas
RF12	Listar trazas
RF13	Visualizar una traza seleccionada

2.2.2. Requisitos no funcionales

Los requisitos no funcionales definen las restricciones del sistema, son propiedades que el sistema debe poseer. Representan las características del producto [32]. En el caso del módulo de Notificación y Monitoreo de Usuarios, se asumen y adoptan las herramientas y tecnologías que propone la Arquitectura para la plataforma de gestión de videojuegos serios Medicando [1].

Requerimientos de *Software*

Se necesitan como requerimientos mínimos:

- **Cliente:** Sistema Operativo *Windows* 7 o superior, *Linux*, navegador web estándar con capacidad de interpretación de *JavaScript* y CSS, *Internet Explorer* 7 o superior, *Firefox* 51.0.1 o superior.
- **Servidor:** Sistema Operativo *Windows* 10 o superior, *Linux* (cualquier distribución), servidor web *Apache* 2.4.9, lenguaje de programación *PHP* 5.4.0 o superior y MySQL 5.6.16 o superior. Se recomienda la utilización del paquete de programas XAMPP en su versión 1.8.3 para Sistemas Windows

y LAMPP para distribuciones de *Linux*, ya que es multiplataforma e incluye *Apache*, *PHP* y MySQL por defecto.

Requerimientos de *Hardware*

Se necesitan como requerimientos mínimos:

- **Cliente:** Procesador DualCore, 1 GB de RAM, 320 MB de disco duro.
- **Servidor:** Procesador i3-4170, 4 GB de RAM, 1 TB de disco duro.

Requerimientos de Apariencia

Tendrá diferentes tonalidades de color guiados por el verde en correspondencia con los colores representativos de la estrategia marcaria de los productos de la UCI destinados a la salud, XAVIA.

Requerimientos de Seguridad

- **Confidencialidad:** Existencia de distintos roles que establecen que la información sólo sea vista por aquellos usuarios que posean los privilegios suficientes. Restricción de la ejecución de acciones a usuarios sin credenciales y verificación de que el usuario esté autentificado antes de realizar alguna acción.
- **Integridad:** Validación de los datos en el servidor para evitar estados inconsistentes. La información manejada por el sistema estará protegida del acceso y divulgación no autorizada. Se debe realizar la confirmación sobre acciones irreversibles como eliminaciones.
- **Disponibilidad:** Los mecanismos utilizados para lograr la seguridad no obstruyen el acceso a la información.

Requerimientos de Diseño e Implentación

Se hace uso del *framework* Yii en su versión 2.0, *NetBeans* 8.2 como IDE de desarrollo, *XAMPP* 1.8.3 como servidor web y MySQL 5.6.16 como SGBD.

2.3. Fase I: Planificación

La fase de planificación es la primera fase que propone la metodoligía XP para el desarrollo de *software*. En esta fase el cliente define el alcance general que tendrá el proyecto. En la misma explica todas sus necesidades mediante la realización de HU y establece las prioridades para cada una de ellas. Posteriormente los programadores realizan la estimación de tiempo de desarrollo basándose en esta información. Las estimaciones realizadas en esta fase son primarias debido a que están basadas en datos de muy alto nivel y estas podrían variar en posteriores iteraciones.

2.3.1. Historias de Usuarios

Las HU son la técnica que utiliza XP para especificar los requisitos de *software* [33]. Una HU es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Las HU son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos (acompañadas de las discusiones con los usuarios y las pruebas de validación). Cada HU debe ser limitada, ésta debería escribirse sobre una nota adhesiva pequeña. Dentro de la metodología XP las HU deben ser escritas por los clientes [34]. A continuación, se describen las HU definidas para llevar a cabo el desarrollo del módulo.

Tabla 2.3. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Administrar notificación
Usuario: Administrardor, Doctor, Paciente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2.0	**Iteración asignada:** 1
Programador responsable: Daylilis Cardoso Roque	
Descripción: Ofrece al administrador una vista para la eliminación de las notificaciones ya leídas que aún se encuentran almacenadas en la base de datos. También permite listar y eliminar las notificaciones realizadas vía correo electrónico. Por otro lado cada usuario es capaz de eliminar sus notificaciones.	
Observaciones: Cada usuario cuenta con una configuración por defecto según el rol que desempeñe en la aplicación. Las notificaciones llegarán en correspondencia además de la vía seleccionada.	
Interfaz: 	

Tabla 2.4. Historia de usuario # 2

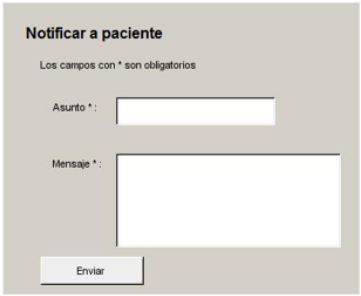
Historia de usuario	
Número: 2	**Nombre:** Enviar notificación
Usuario: Doctor, Paciente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3.0	**Iteración asignada:** 1
Programador responsable: Daylilis Cardoso Roque	
Descripción: De forma dinámica, el sistema envía una notificación a los usuarios cuando se realiza alguna acción en el sistema que le afecte o involucre directamente. Permite al doctor enviar una notificación directamente a un paciente seleccionado. Esta consta con los siguientes campos: • Asunto(Obligatorio) • Mensaje(Obligatorio)	
Observaciones: Se enviarán las notificaciones a cada usuario del sistema en caso de que este desee recibirla y por la vía correspondiente de acuerdo a su configuración personal para recibir las mismas.	
Interfaz: 	

Tabla 2.5. Historia de usuario # 3

Historia de usuario	
Número: 3	**Nombre:** Mostrar notificación
Usuario: Usuario del sistema	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2.0	**Iteración asignada:** 2
Programador responsable: Daylilis Cardoso Roque	
Descripción: Permite a cada usuario observar las notificaciones pendientes que posea mediante una lista desplegable. Además, ofrece la posibilidad de marcar todas las notificaciones como leídas y de mostrar todas las notificaciones que dicho usuario tenga en la base de datos del sistema.	

Continúa en la próxima página

Tabla 2.5. Continuación de la página anterior

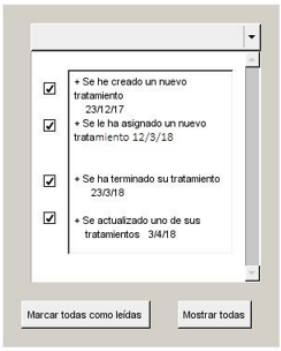
Observaciones: Una vez seleccionada una notificación esta se muestra y se marca automáticamente como leída, por lo que no será listada nuevamente en la lista de notificaciones pendientes.
Interfaz: 

Tabla 2.6. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Configurar notificación
Usuario: Usuario del sistema	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2.0	**Iteración asignada:** 2
Programador responsable: Daylilis Cardoso Roque	
Descripción: Permite que cada usuario pueda configurar por qué canales desea recibir las notifaciones. Los dos canales por los que se pueden recibir las notificaciones son: • Vía correo electrónico. • Vía propio sistema.	
Observaciones: Una vez seleccionada la manera en que se quieran recibir las notificaciones, estas solo llegarán por esta vía.	

Continúa en la próxima página

Tabla 2.6. Continuación de la página anterior

Interfaz:

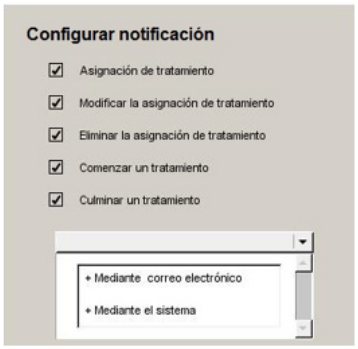


Tabla 2.7. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Administrar trazas
Usuario: Administrardor	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2.5	**Iteración asignada:** 3
Programador responsable: Daylilis Cardoso Roque	
Descripción: Ofrece al administrador una vista en que se listan las trazas generadas por el sistema. Mediante esta vista el administrador puede visualizar el contenido de cada traza generada por el sistema.	
Observaciones: El administrador tiene el control de cada acción que realiza cada usuario en el sistema.	
Interfaz:	

2.3.2. Estimación de esfuerzo por Historia de Usuario

Se realiza la estimación de esfuerzo que arroja cada HU, con el objetivo de obtener un correcto desarrollo del sistema.

Tabla 2.8. Estimación de esfuerzo por HU

	Historias de Usuario	Puntos estimados(Semanas)
1	Administrar notificación	2.0
2	Enviar notificación	3.0
3	Mostrar notificación	2.0
4	Configurar notificación	2.0
5	Administrar trazas	2.5

2.3.3. Desarrollo del plan de iteraciones

Una vez definidas las HU y realizada una previa estimación de esfuerzos, se procede a la planificación de la etapa de implementación del sistema. En este espacio, se crea el plan de iteraciones, donde se especifica la prioridad con que se implementarán las HU organizadas por iteraciones. Teniendo en cuenta el esfuerzo asociado a las mismas y a las prioridades del cliente, se define una versión que sea de valor para este.

2.3.4. Plan de duración de las iteraciones

A continuación, se presenta el plan de duración de las iteraciones. Este plan, tiene como finalidad, mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada iteración como se muestra en la tabla siguiente:

Tabla 2.9. Plan de duración de iteraciones

Iteraciones	No.	Historias de Usuario	Duración(Semanas)
1	1	Administrar notificación	2.0
	2	Enviar notificación	3.0
2	3	Mostrar notificación	2.0
	4	Configurar notificación	2.0
3	5	Administrar trazas	2.5
Total			11.5

Plan de entrega

En el plan de entrega que se plantea a continuación se hace una propuesta de la fecha aproximada en que se harán versiones al sistema al finalizar cada iteración en la fase de implementación.

Tabla 2.10. Plan de duración de iteraciones

Entregable	1ra Iteración	2da Iteración	3ra Iteración
Versión	versión 0.3	versión 0.6	versión 1.0
Fecha	(22-01-2018)	(15-03-2018)	(18-05-2018)

2.4. Fase II: Diseño del sistema

La metodología XP hace especial énfasis en los diseños simples y claros, por lo que en esta fase se define el estilo y patrón arquitectónico utilizado en el desarrollo de la solución propuesta, así como la asignación de responsabilidades basada en los Patrones Generales de Asignación de Responsabilidades (GRASP). La metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC, por lo que en esta fase se describen las clases utilizadas en la solución, se analizan las HU y se descomponen en tareas independientes.

2.4.1. Estilo y Patrón Arquitectónico

Estilo Arquitectónico

Los **estilos arquitectónicos** se definen como: "un conjunto de reglas de diseño que identifica las clases de componentes y conectores que se pueden utilizar para componer en sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo" [35].

Basado en componentes

El estilo de arquitectura basada en componentes, describe un acercamiento al diseño de sistemas como un conjunto de componentes que exponen interfaces bien definidas y que colaboran entre sí para resolver el problema como se muestra en la figura [35].

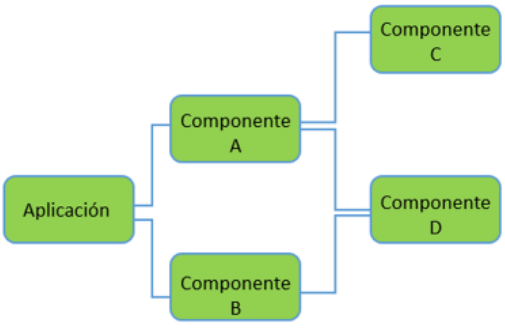


Figura 2.1. Estilo basado en componentes

Este estilo se utiliza para diseñar aplicaciones a partir de componentes individuales, enfatiza la descomposición del sistema en componentes lógicos o funcionales y define una aproximación al diseño a través de componentes que se comunican mediante interfaces que contienen métodos, eventos y propiedades [35]. El uso de este estilo facilita el despliegue, pues permite sustituir un componente por su nueva versión sin afectar a otros componentes o al sistema y favorece la reusabilidad de los componentes independientes del contexto, permitiendo que se empleen en otras aplicaciones y sistemas.

Patrón Arquitectónico

Los **patrones arquitectónicos** se abocan a un problema de aplicación específica dentro de un contexto dado y sujeto a limitaciones y restricciones. El patrón propone una solución arquitectónica que sirve como base para el diseño de la arquitectura [36].

Un patrón arquitectónico expresa una organización estructural para sistemas de software. Se definen sobre aspectos fundamentales de la estructura del sistema, donde se especifican una serie de recomendaciones para organizar los distintos componentes. Yii, en su estructura usa el MVC [1].

MVC

- Separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes como se muestar en la figura 2.2: el **modelo** en el que se administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado y da respuestas a instrucciones de cambiar el estado. La **vista** maneja la visualización de la información. Y el **controlador** que interpreta las acciones del *mouse* (ratón) y el teclado, informando al modelo y/o a las vistas para que cambien según resulte apropiado.

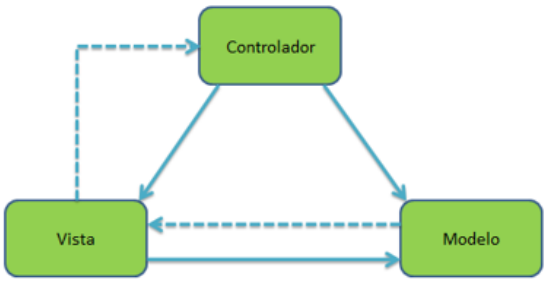


Figura 2.2. Patrón Modelo-Vista-Controlador

La propuesta de solución estará basada igualmente en el estilo y patrón arquitectónico descritos anteriormente. La figura 2.3 muestra la representación arquitectónica del módulo de Notificación y Monitoreo de Usuarios integrados con la plataforma Medicando.

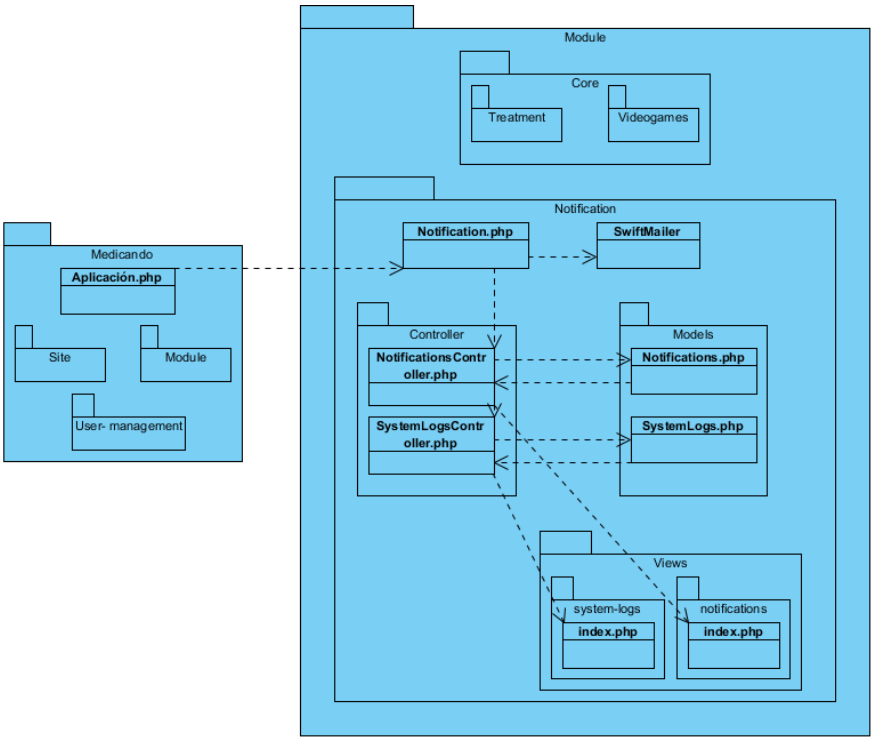


Figura 2.3. Arquitectura del módulo Notificación y Monitoreo de Usuarios

2.4.2. Patrones de diseño

Un **patrón de diseño** se define de la manera siguiente: "Es una mezcla con nombre propio de puntos de vista que contienen la esencia de una solución demostrada para un problema recurrente dentro de cierto contexto de necesidades en competencia"[35]. Son patrones de un nivel de abstracción menor que los patrones de arquitectura. Están, por lo tanto, más próximos a lo que sería el código fuente final. Su uso no se refleja en la estructura global del sistema [37].

Los patrones de diseño trabajan a una escala intermedia y son independientes del lenguaje de programación que se utilice. Su aplicación no tiene efectos en la estructura fundamental del sistema (arquitectura), pero puede tener una fuerte influencia sobre la arquitectura de un subsistema. Definen un esquema de refinamiento de los subsistemas o componentes dentro de un sistema, o las relaciones entre estos [37].

Para la asignación general de responsabilidades en el software existen patrones denominados GRASP, que describen los principios fundamentales de asignación de responsabilidades a objetos, expresadas como patrones. Seguidamente se exponen algunos patrones utilizados dentro del contexto de los módulos a implementar:

- ***Experto*:** Asigna la responsabilidad a la clase que tiene la información necesaria para cumplir la responsabilidad [38]. En Yii toda la información de los datos la manejan las clases modelos, pues son estas las que acceden a la base de dato en respuesta de una petición de las clases controladoras para satisfacer un evento iniciado por el usuario.
- ***Controlador*:** Asignar la responsabilidad de manejar los eventos del sistema a una clase. Un evento de entrada al sistema es algún evento desde algún actor externo. La clase controlador es responsable de decidir qué hacer con el evento. El controlador actúa como una fachada entre la interfaz y la aplicación [38]. Yii tiene bien definidas las clases controladoras dentro de su patrón MVC, las cuales se encargan de delegar responsabilidades, en respuesta a alguna acción del usuario, a las clases modelo e interfaces.
- ***Creador*:** Define la clase que debe tener la responsabilidad de crear una nueva instancia de otra clase. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Se evidencia en la clase controladora *NotificationsController*. Esta clase se encarga de modificar la configuración de las notificaciones, se ofrece así una alta cohesión. Permite, de esta forma, establecer un bajo acoplamiento, lo que propicia menos dependencias funcionales y ofrece mejores oportunidades de reutilización.
- ***Active Record*:** Define una forma de acceder a los datos de una base de datos y convertir las filas de una tabla en objetos. Yii, implementa el patrón de persistencia *ActiveRecord*, que facilita el control de la información almacenada en la base de datos. Se trata de una clase que se encarga de implementar todas las operaciones de consulta y modificación de una tabla concreta de la base de datos. De esta forma, la aplicación delega el trabajo con SQL, a la capa de componentes *ActiveRecord* que maneja Yii.

2.4.3. Tarjetas CRC

A continuación, las HU son evaluadas para dividirlas en tareas, cada tarea representa una característica distinta del sistema y se puede diseñar una prueba de unidad que verifique cada tarea, estas tareas se representan por medio de las tarjetas CRC.

Las tarjetas CRC permitirán desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología asentada en objetos, permiten también que el equipo completo contribuya en la tarea del diseño. Estas tarjetas identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades), cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o que colaboran con ella. Las tarjetas CRC son el único trabajo de diseño que se genera como parte del proceso de XP [31].

Las principales características de las tarjetas CRC son:

- Identificación de clases y asociaciones que participan del diseño del sistema.

- Obtención de las responsabilidades que debe cumplir cada clase.
- Establecimiento de cómo una clase colabora con otras clases para cumplir con sus responsabilidades.

Seguidamente se realiza un representación de las principales tarjetas CRC diseñadas para llevar a cabo la implementación del módulo propuesto.

Tabla 2.11. Tarjeta CRC # 1

Tarjeta CRC	
Clase: Notification	
Responsabilidad	**Colaboración**
• Representar los datos persistentes en la base de datos correspondientes a las notificaciones del sistema. Siendo una clase Modelo o también conocida como Entidad. • getUserFrom() Devuelve el id del usuario que generó la notificación, al realizar una acción en el sistema. • getUserTo() Devuelve el id del usuario al cual va dirigido la notificación generada.	• Notification • Videogame • Treatment • Doctor • Patient

Tabla 2.12. Tarjeta CRC # 2

Tarjeta CRC	
Clase: NotificationsController	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla 2.12. Continuación de la página anterior

• actionIndex(): Funcionalidad que muestra la vista donde se listan las Notificaciones. • actionCheckReaded(id): Funcionalidad que comprueba si el estado de una notificación es *Leído*, en caso contrario cambia el estado de la misma a *Leído*. • actionCheckAll(): Funcionalidad que permite actualizar el estado de todas las notificaciones que se tienen registradas en la base de datos. • actionDeleteSelected(): Funcionalidad que permite eliminar de manera simultánea un grupo de notificaciones previamente seleccionadas.	• Notification • Videogame • Treatment • Doctor • Patient

Tabla 2.13. Tarjeta CRC # 3

Tarjeta CRC	
Clase: SystemLogs	
Responsabilidad	**Colaboración**
• Representar los datos persistentes en la base de datos correspondientes a las trazas del sistema. Es una clase Modelo o también conocida como Entidad. • getUser() Devuelve el id del usuario que realizó la acción y genera una traza del sistema.	• Notification • Videogame • Treatment • Doctor • Patient • Especiality

Tabla 2.14. Tarjeta CRC # 4

Tarjeta CRC	
Clase: SystemLogsController	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla 2.14. Continuación de la página anterior

• actionIndex(): Funcionalidad que muestra la vista donde se listan las Trazas del sistema. • findModel(id): Funcionalidad que permite buscar los datos de una traza del sistema.	• Notification • Videogame • Treatment • Doctor • Patient • Especiality

2.5. Conclusiones del capítulo

En este capítulo se han abordado los aspectos referentes a la concepción del producto a desarrollar y sus características funcionales. Esto permitió arribar a las siguientes conclusiones:

- A partir de la definición de las HU se pudieron identificar las principales funcionalidades a desarrollar en correspondencia con el módulo de Notificación y Monitoreo de Usuarios propuesto.
- La estimación del tiempo para la implementación de las HU definidas, permitió calcular una entrega final del producto en cuatro meses aproximadamente.
- Fue posible identificar los patrones GRASP para el desarrollo de la solución, así como el patrón de persistencia *ActiveRecord* que implementa el *framework* Yii.
- Se exponen los artefactos generados en las fases de Planificación y Diseño que establece la metodología XP.

CAPÍTULO 3

Pruebas al Sistema

En el presente capítulo se detallan las iteraciones realizadas durante la etapa de construcción del módulo propuesto, además se exponen las tareas de ingeniería generadas para cada HU que fueron definidas, así como las pruebas de aceptación planificadas para el sistema. De esta forma es obtenido un producto funcional probado y listo para entregar al cliente al final de cada iteración como propone XP.

3.1. Fase III: Desarrollo

En esta fase, XP plantea que las HU seleccionadas para ser implementadas se realizan durante el transcurso de la iteración a la que pertenecen. Por estas razones, se lleva a cabo una revisión del plan de iteraciones y se modifican en caso de ser necesario. Como parte de este plan se descomponen las HU en tareas de ingeniería [29].

3.1.1. Tareas de ingeniería

Tienen como objetivo definir cada una de las actividades que dan cumplimiento a las HU, de forma tal que se entienda lo que el sistema tiene que hacer y facilite su construcción. Pueden estar descritas por un lenguaje técnico y no ser necesariamente entendibles por el cliente [33]. Se describen algunas de las tareas de ingeniería correspondientes a las HU del sistema, se definen en correspondencia con las iteraciones definidas como se muestra a continuación.

Tareas de ingeniería para la Iteración I

Para la primera iteración, se definieron un total de siete tareas de ingeniería, cada una desglosada a partir de la HU correspondiente. Se describen algunas de las tareas definidas, el resto pueden ser consultadas en los anexos (Ver Anexo A.3).

- ***HU Administrar notificaciones***

Tabla 3.1. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** 1
Nombre de la tarea: Listar notificaciones	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 15 de octubre de 2017	**Fecha de fin:** 1 de noviembre de 2017
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad que permite obtener una lista con todas las notificaciones generadas por el sistema o por algún doctor en específico. En caso que el usuario registrado sea el administrador del sistema la lista mostrará todas las notificaciones del sistema. En caso que el usuario registrado sea doctor o paciente, la lista mostrará las notificaciones pendientes relacionadas solo con el este usuario.	

Tabla 3.2. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** 1
Nombre de la tarea: Eliminación de notificaciones de manera simultánea	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 2 de noviembre de 2017	**Fecha de fin:** 16 de noviembre de 2017
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad, que permite eliminar un conjunto de notificaciones previamente seleccionadas. Estas notificaciones pueden ser seleccionadas por cualquier usuario del sistema.	

- ***HU Enviar notificación***

Tabla 3.3. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 2
Nombre de la tarea: Notificación a enviar	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha de inicio: 2 de diciembre de 2017	**Fecha de fin:** 16 de diciembre de 2017
Programador responsable: Daylilis Cardoso Roque	

Continúa en la próxima página

Tabla 3.3. Continuación de la página anterior

Descripción: Se diseña un procedimiento lógico para elaborar el cuerpo de las notificaciones. Para ello, se tiene en cuenta el usuario que realiza la acción, los usuarios implicados y la acción realizada. Luego, este mecanismo se utiliza para implementar el desarrollo del cuerpo de la notificación, o sea, desarrollar el modelo o entidad.

Tabla 3.4. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 2
Nombre de la tarea: Comprobación de destinatarios y enviar notificación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha de inicio: 17 de diciembre de 2017	**Fecha de fin:** 5 de enero de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad que permita chequear si cada destinatario de la notificación generada, tiene configurado el tipo de notificación en cuestión. En caso positivo, se envía la notificación por las vías que se encuentren definidas por el destinatario. En caso contrario, la notificación no se envía, ni se almacena en la base de datos.	

Tareas de ingeniería para la Iteración II

Para la segunda iteración, se definieron un total de cinco tareas de ingeniería, cada una desglosada a partir de la HU correspondiente. Se describen algunas de las tareas definidas, el resto pueden ser consultadas en los anexos (Ver Anexo A.3).

- ***HU Mostrar notificación***

Tabla 3.5. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 3
Nombre de la tarea: Mostrar las notificaciones pendientes de un usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 22 de diciembre de 2017	**Fecha de fin:** 29 de enero de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad que permita mostrar los detalles de las notificaciones asociadas a un usuario registrado.	

Tabla 3.6. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 3
Nombre de la tarea: Vistas para mostrar la información de la notificación seleccionada	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 30 de enero de 2018	**Fecha de fin:** 6 de febrero de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se desarrolla una vista para mostrar la información recopilada sobre las notificaciones en cuestión. Además, se ofrece la posibilidad de eliminar dicha notificación de la base de datos.	

- ***HU Configurar notificación***

Tabla 3.7. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 4
Nombre de la tarea: Configuración personal del usuario	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha de inicio: 23 de febrero de 2018	**Fecha de fin:** 4 de marzo de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad que permita a cada usuario escoger por que canal de envío quiere recibir las notificaciones, estos canales pueden ser: • Vía correo electrónico. • Vía alerta de sistema.	

Tareas de ingeniería para la Iteración III

Para la tercera iteración, se definieron un total de tres tareas de ingeniería, cada una desglosada a partir de la HU correspondiente. Se describen algunas de las tareas definidas, el resto pueden ser consultadas en los anexos (Ver Anexo A.3).

- ***HU Administrar trazas***

Tabla 3.8. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 5
Nombre de la tarea: Generar trazas	

Continúa en la próxima página

Tabla 3.8. Continuación de la página anterior

Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha de inicio: 16 de marzo de 2018	**Fecha de fin:** 15 de abril de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se diseña un procedimiento lógico que permite generar trazas cada vez que un usuario del sistema realice alguna acción señalada como crítica por su importancia en el negocio.	

Tabla 3.9. Tarea de ingeniería # 9

Tarea	
Número de tarea: 9	**Número de Historia de usuario:** 5
Nombre de la tarea: Listar trazas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 16 de abril de 2018	**Fecha de fin:** 30 de abril de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad que permite obtener una lista con todas las trazas generadas por el sistema, esta lista solo puede ser consultada por el administrador del sistema.	

Con las tareas de ingeniería definidas, se hace necesario establecer un conjunto de pruebas para comprobar la calidad de la solución implementada. Luego, se analizan estos casos de prueba y se ejecutan, lo que permite medir el nivel de cumplimiento con los objetivos de implementación trazados y el nivel de satisfacción del cliente.

3.2. Fase IV: Pruebas

Uno de los pilares de XP es el proceso de pruebas ya que anima a probar constantemente o tanto como sea posible [39]. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente [40].

Las características clave de las pruebas en XP son [32]:

- Desarrollo previamente probado.
- Desarrollo de pruebas incremental a partir de los escenarios.
- Participación del usuario en el desarrollo de las pruebas y en la validación.
- El uso de bancos de pruebas automatizados.

3.2.1. Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP, todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código [29].

Estas pruebas también son llamadas pruebas modulares ya que permiten determinar si un módulo del programa está listo y correctamente terminado. El objetivo fundamental de las pruebas unitarias es asegurar el correcto funcionamiento de las interfaces o flujo de datos entre componentes [41].

En el Módulo Notificación y Monitoreo de Usuarios las pruebas unitarias se realizaron al unísono con las pruebas de aceptación, debido a que en la realización de las mismas los casos de pruebas son diseñados para cada una de las tareas de ingienería definidas, que a su vez constituyen la unidad atómica de las HU, por lo que la realización de todos los casos de pruebas definidos para una HU constituye la realización de la prueba a esa unidad.

Para realizar esta prueba se diseñó el grafo de flujo para cada una de las operaciones contenidas en los componentes implementados. Posteriormente se calculó la complejidad ciclomática para cada grafo de flujo, esto arrojó un valor promedio de tres puntos (Ver Anexo A.4.1). Este valor puntual indica que se disponen a lo sumo de tres caminos lógicos linealmente independientes para la ejecución de escenarios en sus métodos, por lo que se deben realizar tres casos de pruebas para esta funcionalidad (Ver 3.2.3). Demostrando el grado de optimización de las funcionalidades desarrolladas como parte de la arquitectura base para el desarrollo del módulo. Además, este resultado denota la aplicabilidad en la solución arquitectónica propuesta de otros atributos de calidad, tales como: extensibilidad, fiabilidad y mantenibilidad.

3.2.2. Pruebas de integración

Las pruebas de integración son una técnica sistemática para construir la arquitectura del *software* mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz [35]. La prueba de *integración descendente* es un enfoque incremental a la construcción de la arquitectura de *software*. Los módulos se integran al moverse hacia abajo a través de la jerarquía de control, comenzando con el módulo de control principal (programa principal). Los módulos subordinados al módulo de control principal se incorporan en la estructura [35].

La integración *primero en anchura* incorpora todos los componentes directamente subordinados en cada nivel, y se mueve horizontalmente a través de la estructura. Los controladores Doctor, Paciente, Especialidad, Tratamiento y Videojuegos se integran primero al componente Controladores, debido a que en cada uno de los mismos es donde el usuario realiza acciones declaradas de importancia para el negocio, este componente se integra al Módulo de Notificación y Monitoreo de Usuarios, que se encarga de generar notificaciones a los usuarios correspondientes y trazas solo visibles por el administrador del sistema, este a su vez al siguiente nivel de control Aplicación (Plataforma Medicando) como muestra la figura 3.1.

Una vez concluido el proceso de integración del módulo Notificación y Monitoreo de Usuarios, se pudo comprobar que se integra correctamente a la plataforma y puede ser utilizado tanto por los subsistemas con

que cuenta Medicando como por los demás módulos de la plataforma.

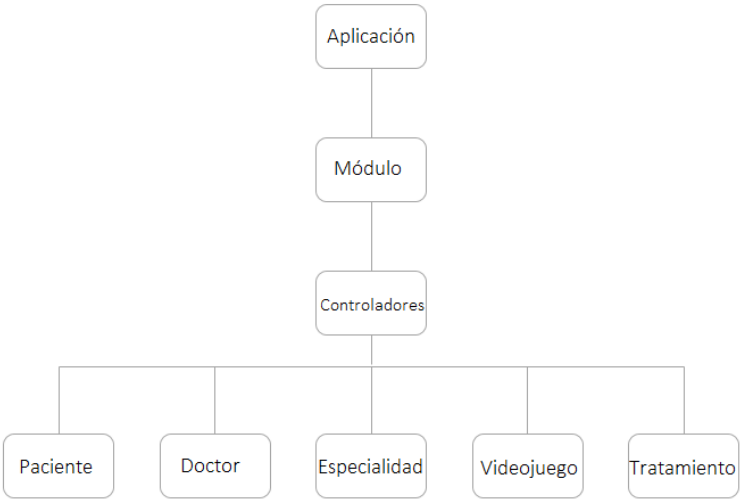


Figura 3.1. Integración del módulo Notificación y Monitoreo de Usuarios

3.2.3. Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. Las pruebas de aceptación son consideradas como "pruebas de caja negra" (Ver Anexo A.4.2). Los clientes son responsables de verificar que los resultados de éstas pruebas sean correctos [29].

Las pruebas de aceptación se realizan sobre el producto terminado [42]. A continuación se definen un grupo de Pruebas de Aceptación que permitirán comprobar la correcta implementación de las funcionalidades propuestas para el módulo de Notificación y Monitoreo de Usuarios.

Pruebas de aceptación para la Iteración I

Para la primera iteración, se definieron un total de siete casos de pruebas de aceptación. Todas enfocadas a evaluar la implementación de algunas funcionalidades del módulo. Se describen algunas de las pruebas realizadas, el resto pueden ser consultadas en los anexos (Ver Anexo A.4.2) de la investigación.

Tabla 3.10. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Listar notificaciones	
Descripción: Prueba de la funcionalidad de listar notificaciones	

Continúa en la próxima página

Tabla 3.10. Continuación de la página anterior

Condiciones de ejecución: • El usuario deberá estar autenticado con el rol de Administrador.
Pasos de ejecución: 1. El Administrador desde la vista inicial del sistema selecciona ir a la vista de notificaciones. 2. Como respuesta, del sistema, se direcciona hacia la vista de notificaciones y muestra todas las notificaciones que han sido generadas por el sistema.
Resultados esperados: Satisfactorio

Tabla 3.11. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU1_P2	**Historia de usuario:** 1
Nombre: Eliminación de notificaciones de manera simultáneas	
Descripción: Prueba de la funcionalidad de eliminación de notificaciones de manera simultáneas	
Condiciones de ejecución: • El usuario deberá estar autenticado.	
Pasos de ejecución: 1. El usuario entra a la vista Notificaciones, donde se listan todas las notificaciones que este tenga. 2. Se seleccionan todos las notificaciones correspondientes a la primera página de notificaciones listadas. 3. Se marca la opción *Borrar seleccionados*. 4. Se muestra una alerta indicando que se eliminaron correctamente las notificaciones seleccionadas.	
Resultados esperados: Satisfactorio	

Tabla 3.12. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU1_P3	**Historia de usuario:** 1
Nombre: Eliminación de notificaciones de manera simultáneas sin ser seleccionadas	
Descripción: Prueba para la funcionalidad administrar notificaciones (eliminar notificaciones de manera simultáneas). En este caso no se selecciona ningún elemento a eliminar.	
Condiciones de ejecución: • El usuario deberá estar autenticado.	

Continúa en la próxima página

Tabla 3.12. Continuación de la página anterior

Pasos de ejecución: 1. El usuario entra a la vista Notificaciones, donde se listan todas las notificaciones que este tenga. 2. Se marca la opción *Borrar seleccionados*, sin seleccionar ninguna notificación. 3. Se muestra una alerta de error indicando que no se ha seleccionado ningún elemento para la operación solicitada.
Resultados esperados: Satisfactorio

Tabla 3.13. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU2_P5	**Historia de usuario:** 2
Nombre: Enviar notificación.	
Descripción: Prueba para la funcionalidad enviar notificación activada al crear un nuevo tratamiento.	
Condiciones de ejecución: • El usuario deberá estar autenticado y contar con el rol de Doctor. • En el sistema deben estar registrados al menos dos usuarios con rol de Doctor. • El usuario implicado, debe tener configurado, recibir una notificación para la acción: Creación de tratamiento.	
Pasos de ejecución: 1. El doctor autenticado desde la vista de tratamientos desea insertar un nuevo tratamiento. 2. Como respuesta, el sistema; se direcciona hacia una vista en la que se muestra un formulario con los datos a llenar del nuevo tratamiento que se desea crear. 3. El doctor selecciona la opción de crear 4. Como respuesta, el sistema; se direcciona hacia una vista en la que se muestra un formulario con los datos del nuevo tratamiento creado. En correspondencia se envía automáticamente un correo electrónico o/y un mensaje del sistema a los demás Doctores registrados en el sistema, dichos mensajes contienen los detalles de la acción realizada.	
Resultados esperados: Satisfactorio	

Pruebas de aceptación para la Iteración II

Para la segunda iteración, se definieron un total de cinco casos de pruebas de aceptación. Todas enfocadas a evaluar la implementación de algunas funcionalidades del módulo. Se describen algunas de las pruebas realizadas, el resto pueden ser consultadas en los anexos (Ver Anexo A.4.2) de la investigación.

Tabla 3.14. Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: Mostrar notificaciones pendientes.	
Descripción: Prueba para la funcionalidad mostrar notificaciones pendientes.	
Condiciones de ejecución: • El usuario deberá estar autenticado. • El usuario debe tener al menos una notificación pendiente a visualizar.	
Pasos de ejecución: 1. El usuario desde la vista inicial del sistema, presiona el ícono de notificaciones pendientes (campana) que aparece en la esquina superior del banner. 2. Como respuesta, el sistema; muestra en una lista desplegable todas las notificaciones pendientes que dicho usuario tenga en el sistema, además brinda la posibilidad de seleccionar las funcionalidades: *Ver todas y Marcar todas como leídas*.	
Resultados esperados: Satisfactorio	

Tabla 3.15. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU3_P2	**Historia de usuario:** 3
Nombre: Visualizar la información de la notificación seleccionada	
Descripción: Prueba para la funcionalidad visualizar la información de la notificación seleccionada.	
Condiciones de ejecución: • El usuario deberá estar autenticado. • El usuario debe tener al menos una notificación en el sistema.	
Pasos de ejecución: 1. El usuario desde la vista Notificaciones, donde se listan todas las notificaciones que este tenga, selecciona ver todos los datos de una notificación seleccionada. 2. Como respuesta, el sistema; se direcciona hacia una vista en la que se muestra los datos de la notificación seleccionada mediante una tabla. Dicha vista ofrece al usuario la posibilidad de eliminar dicha notificación de la base de datos.	
Resultados esperados: Satisfactorio	

Tabla 3.16. Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU3_P5	**Historia de usuario:** 4
Nombre: Configuración personal del usuario.	

Continúa en la próxima página

Tabla 3.16. Continuación de la página anterior

Descripción: Prueba para la funcionalidad configuración personal del usuario.
Condiciones de ejecución: • El usuario deberá estar autenticado.
Pasos de ejecución: 1. El usuario accede a la configuración de su perfil. 2. Como respuesta, el sistema; muestra todas las configuraciones posibles del usuario. 3. El usuario selecciona la configuración de notificaciones. 4. Como respuesta, el sistema; muestra en una lista desplegable las opciones de recibo de notificaciones que extisten: *Vía correo electrónico y A travéz del propio sistema* y brinda la opción de *Guardar y Actualizar* los cambios. 5. El usuario marca todas las opciones de notificación vía correo electrónica y ninguna a través del sistema. 6. El usuario selecciona la opción *Guardar*. 7. Se muestra una alerta indicando que se guardó correctamente la configuración del usuario.
Resultados esperados: Satisfactorio

Pruebas de aceptación para la Iteración III

Para la tercera iteración, se definieron un total de dos casos de pruebas de aceptación. Todos enfocados a evaluar la implementación de algunas funcionalidades del módulo, relacionadas al monitoreo de usuarios.

Tabla 3.17. Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU5_P1	**Historia de usuario:** 5
Nombre: Listar trazas	
Descripción: Prueba de la funcionalidad de listar trazas	
Condiciones de ejecución: • El usuario deberá estar autenticado con el rol de Administrador.	
Pasos de ejecución: 1. El Administrador desde la vista inicial del sistema selecciona ir a la vista de trazas. 2. Como respuesta, el sistema, direcciona hacia la vista de trazas y muestra todas las trazas generadas por el sistema.	
Resultados esperados: Satisfactorio	

Tabla 3.18. Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU5_P2	**Historia de usuario:** 5
Nombre: Visualizar la información de la traza seleccionada	
Descripción: Prueba para la funcionalidad visualizar la información de la traza seleccionada.	
Condiciones de ejecución: • El usuario deberá estar autenticado con el rol de Administrador.	
Pasos de ejecución: 1. El Administrador desde la vista Trazas, donde se listan todas las trazas que se hayan generado, selecciona ver todos los datos de una traza seleccionada. 2. Como respuesta, el sistema; se direcciona hacia una vista en la que se muestra los datos de la trazas seleccionada mediante una tabla.	
Resultados esperados: Satisfactorio	

3.2.4. Análisis de las pruebas de aceptación

Se desarrollaron un total de catorce casos de pruebas de aceptación. Estas pruebas fueron realizada de forma organizada, por cada iteración definida. A continuación, se muestra en gráficas, los porcientos de satisfacción alcanzados en cada iteración.

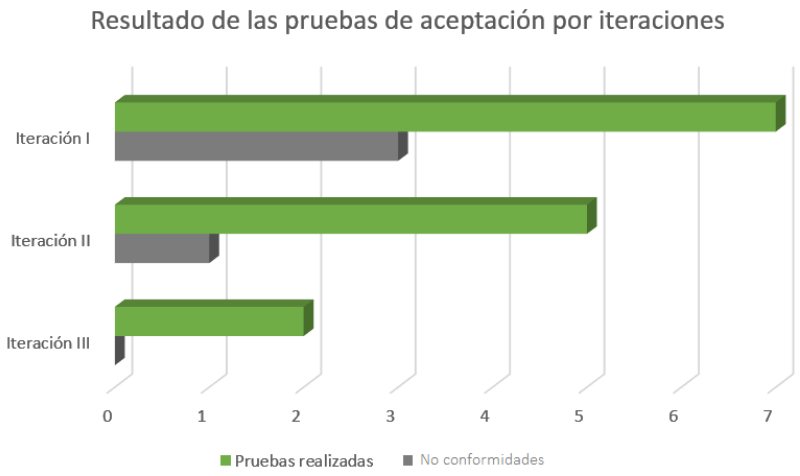


Figura 3.2. Resultados de las pruebas de aceptación

Como se puede observar en la figura 3.2 , en la primera iteración se realizaron un total de 7 pruebas, de ellas 4 alcanzaron el nivel de satisfacción esperado, obteniendo un 58 % de satisfacción, una de las pruebas, detectó que se notificaba al paciente de forma directa, sin validar que los campos estuviesen vacíos. Lo que permitió corregir la falla, e incorporar la seguridad de esta operación en el sistema. En la segunda iteración se realizaron un total de 5 pruebas de ellas 4 alcanzaron el nivel de satisfacción esperado, alcanzando un

80 % de satisfacción, una de las pruebas, detectó que una vez que el usuario selecciona la opción *Marcar todas como leídas* el sistema no elimina dichas notificaciones de las lista de notificaciones pendientes de dicho usuario. Lo que permitió corregir el error en el menor tiempo posible. En tanto en la tercera iteración se realizaron 2 pruebas, donde todas son evaluadas de resultado satisfactorio. Con estas comprobaciones, se obtiene un 100 % de satisfacción en la iteración final del producto, comprobando el correcto funcionamiento de las funcionalidades implementadas.

3.3. Conclusiones del capítulo

En este capítulo se especificó el proceso de implementación del sistema a partir del desglose de las HU en tareas de ingeniería, lo que permitió especificar los procedimientos necesarios para dar cumplimiento a cada una de ellas. Además se definieron y aplicaron pruebas unitarias, pruebas de integración y pruebas de aceptación a las HU, el módulo y las funcionalidades del mismo respectivamente. Estas pruebas permitieron detectar fallas en el sistema y corregirla, lo que posibilitó mejorar la operabilidad del mismo.

Conclusiones

En el presente trabajo de diploma se desarrolló el Módulo Notificación y Monitoreo de usuarios para la plataforma medicando, el que permite una retroalimentación entre paciente-especialista mediante la generación de notificaciones y el monitoreo de usuarios mediante la generación de trazas, dando cumplimiento al objetivo propuesto al inicio de la investigación. Adicionalmente, se concluye que:

- El módulo implementado ofrece una interfaz que permite la comunicación con los subsistemas principales de Medicando (Tratamiento, Videojuegos y Especialidad) y facilita su uso en otros subsistemas o módulos que generen notificaciones y trazas.
- El componente de trazas implementado permite al administrador del sistema monitorear la actividad de los usuarios, por lo que se puede emplear como una herramienta de análisis ante los fallos que puedan ocurrir.
- Se puede destacar la utilización de la metodología XP como eficiente para guiar el proceso ingenieril en el desarrollo de módulos para aplicaciones web.

Recomendaciones

A partir de los resultados obtenidos se recomienda:

- Extender el uso del componente de notificaciones a todos los módulos de la plataforma Medicando que lo requieran.
- Utilizar bibliotecas *JavaScript* que permitan el uso de *socket* para realizar el envío de las notificaciones en tiempo real.

Glosario

Apache Server Servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. 4

IDE Integrated Development Enviroment, traducido al español Ambiente Integrado de Desarrollo es un entorno de programación, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. 10, 13, 19

JavaScript Lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. 18, 48

PHP Lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. 4, 9–11, 13, 15, 16, 18, 19

XAVIA Marca registrada por la Universidad de las Ciencias Informáticas para las soluciones informáticas en el sector de Salud, donde son implementadas en su mayoría sobre plataformas de software libre o código abierto. 19

Acrónimos

CRC Contenido, Responsabilidad y Colaboración. 3, 25, 28, 29
CSS *Cascade Style Sheet*. 10, 18

GRASP Patrones Generales de Asignación de Responsabilidades. 25, 27, 32

HTML *hypertext markup language*. 10
HTTP *hypertext transfer protocol*. 10
HU Historia de Usuario. 3, 14, 19, 20, 24, 25, 28, 32, 33, 35, 38, 39, 46

MVC Modelo-Vista-Controlador. 9, 26, 28
MySQL Sistema de Gestión de Bases de Datos relacional. 4, 11, 13, 18, 19

SGBD Sistema Gestor de Base de Datos. 10, 13, 19

UCI Universidad de Ciencias Informáticas. 1, 19

VERTEX Centro de Entornos Interactivos 3D. 1

XP eXtreme Programming. 3, 11–13, 19, 20, 25, 28, 32, 33, 37, 38, 47

Referencias bibliográficas

[1] Gabriel Lázaro García Díaz. «Arquitectura para la plataforma de gestion de videojuegos serios Medicando». Tesis doct. Universidad de las Ciencias Informaticas Vertex, Entornos Interactivos 3D, Facultad 4, 2017 (vid. págs. 4, 9, 18, 26).

[2] Ernesto Gutierrez Ramos. «Plataforma web para la gestión de videojuegos serios de navegador con fines terapéuticos (Medicando)». Tesis doct. Universidad de las Ciencias Informáticas Centro de Entornos Interactivos 3D (VERTEX) Facultad 5, 2015 (vid. pág. 4).

[3] Real Academia Española. *Diccionario de la Lengua Española*. Felipe IV, 4 - 28014 Madrid. 2018. URL: http://dle.rae.es/?id=Qec6jfG (vid. pág. 4).

[4] Julio Escobar y col. «Gestión de discusiones y notificaciones en el espacio virtual como base para el trabajo con comunidades de práctica en el contexto educativo». En: (2015) (vid. págs. 5, 7).

[5] Lianny Hernández De la Paz y Alexei Darias Jojorina. «Desarrollo del componente de notificaciones del Sistema de Información Hospitalaria alas HIS.» Tesis doct. Universidad de las Ciencias Informáticas, 2013. URL: hhttp://bibliodoc.uci.cu/RDigitales/2013/%20octubre/1/TD_06388_13.pdfi. (vid. pág. 5).

[6] Verónica Anzures. *Monitoreo de usuarios reales*. Menta Network. 2018. URL: https://www.menta.com.mx/servicios/monitoreo-de-usuarios-reales (vid. pág. 5).

[7] 1&1. *Ficheros log: Toda la información de registro en un archivo*. 2017. URL: https://www.1and1.es/digitalguide/online-marketing/analisis-web/el-log-el-archivo-de-registro-de-procesos-informaticos/ (vid. pág. 6).

[8] Facebook. «Aspectos básicos y configuración de las notificaciones». En: (2017) (vid. págs. 6, 7).

[9] Inc. Twitter. «Información sobre la cronología de Notificaciones». En: (2017) (vid. pág. 7).

[10] Elisa Piva. *ASCP. Evaluation of Effectiveness of a Computerized Notification System for Reporting Critical Values.* 2013. URL: http://ajcp.ascpjournals.org/content/131/3/432.long. (vid. pág. 7).

[11] QSOFT. «Software Salus.Dossier de Descripción. Documento en formato PDF». En: (2009) (vid. pág. 8).

[12] GitHub. *SwiftMailer Extension for Yii 2*. 2018 GitHub, Inc. 2018. URL: https://github.com/yiisoft/yii2%20swiftmailer (vid. pág. 8).

[13] Stack Exchange. *How to use the swiftMailer in Yii2*. 2018 Stack Exchange Inc. 2018. URL: https://stackoverflow.com/questions/24995620/how%20to%20use%20the%20swiftmailer%20in%20yii2 (vid. pág. 8).

[14] Carlos Delgado. *Top 10 : Best notification libraries and plugins for Javascript and jQuery*. Our Code World. 2017. URL: https://ourcodeworld.com/articles/read/52/top10%20best%20notification%20libraries%20and%20plugins%20for%20javascript%20and%20jquery (vid. pág. 9).

[15] Marcelo Dolce. *IZITOAST*. 2016. URL: http://izitoast.marcelodolce.com (vid. pág. 9).

[16] Jordisan. *¿Qué es un framework?* 2007. URL: https://jordisan.net/blog/2006/que-es-un-framework/ (vid. pág. 9).

[17] Mark Safronov y Jeffrey Winesett. *Web Application Development with Yii 2 and PHP*. Ed. por Usha Iyer. Packt Publishing Ltd, 2014 (vid. pág. 9).

[18] J. Winesett. *Agile Web Application Development with Yii 1.1 and PHP5*. Ed. por Aanchal Kumar. Packt Publishing Ltd., 2010 (vid. pág. 9).

[19] Qiang Xue y col. «The Definitive Guide to Yii 2.0». En: *Yii Software LLC*. (2014) (vid. pág. 10).

[20] MkDocs. *Yii2 Framework*. 2018. URL: https://yii2-framework.readthedocs.io/en/stable/guide-es/intro-yii/ (vid. pág. 10).

[21] Netbeans.org. *NetBeans IDE - Overview*. 2018. URL: https://netbeans.org/features/index.html (vid. pág. 10).

[22] *Sistemas de Gestión de Bases de datos y SIG*. 2013. URL: http://www.um.es/geograf/sigmur/sigpdf/temario_9.pdf (vid. pág. 10).

[23] D. Pecos. *PostGreSQL vs. MySQL*. geekWare. 2002. URL: http://danielpecos.com/documents/postgresql-vs-mysql/#AEN11 (vid. pág. 10).

[24] Ian Gilfillan. *La Biblia MySQL*. Anaya Multimedia, 2008 (vid. pág. 10).

[25] Softonic. *XAMPP*. 2018. URL: https://xampp-windows.softonic.com/ (vid. pág. 11).

[26] Business School. *¿Qué son las metodologías de desarrollo de software OBS Business School?* Universidad de Barcelona. 2016. URL: https://www.obs-edu.com/int/blog-project-management/metodolog%C3%ADa-agile/que-son-las-metodologias-de-desarrollo-de-software (vid. pág. 11).

[27] 4Soluciones. *¿Qué es una metodología de desarrollo de software?* 2013. URL: http://www.4rsoluciones.com/blog/una-metodologia-desarrollo-software/ (vid. pág. 11).

[28] Yolanda Borja López. «Metodología Ágil de Desarrollo de Software ? XP». En: *ESPE, MEVAST* (2014) (vid. pág. 11).

[29] Ing. José Joskowicz. «Reglas y Prácticas en eXtreme Programming». Tesis de mtría. Universidad de Vigo, 2008 (vid. págs. 11, 33, 38, 39).

[30] Janeth Rozo Nader. «Metodología de Desarrollo de Software: MBM (Metodologia Basada en Modelos)». Tesis de mtría. INGENIARE, Universidad Libre-Barranquilla, 2014 (vid. pág. 11).

[31] Rosado Gómez Alveiro, Quintero Duarte Alexander y Meneses Guevara Cesar Daniel. «DESARROLLO ÁGIL DE SOFTWARE APLICANDO PROGRAMACIÓN EXTREMA». En: *Revista Ingenio* (2012) (vid. págs. 11, 28).

[32] IAN SOMMERVILLE. *Ingeniería del software*. Ed. por Miguel Martín-Romo. PEARSON EDUCACIÓN, S.A., Madrid, 2005 (vid. págs. 17, 18, 37).

[33] Juan Gabriel Valdés Díaz. «Modulos de visualización de la información paciente -especialista para la plataforma de gestion de videojuegos Medicando». Tesis doct. U niversidad de las C iencias I nformaticas Entornos Interactivos 3D Vertex, Facultad 5, 2016 (vid. págs. 20, 33).

[34] Juan Palacio. «Historias de usuario». En: *Creative Commons Attribution Share* (2013) (vid. pág. 20).

[35] Ph.D. Roger S. Pressman. *Ingeniería del software un Enfoque Práctico*. Ed. por María Teresa Zapata Terrazas. McGRAW-HILL INTERAMERICANA EDITORES, S.A. DE C.V, 2010 (vid. págs. 25-27, 38).

[36] Craig Larman. *UML y patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Ed. por David Fayerman Aragón. Pearson Educacion, 2003 (vid. pág. 26).

[37] Eva Lleonart Martín. «PATRONES». Tesis de mtría. Facultad de Informática - Universidad Politécnica de Valencia, 2014 (vid. pág. 27).

[38] Macario Polo Usaola. «Patrones GRASP». En: (2015) (vid. pág. 28).

[39] Pete McBreen. *Questioning Extreme Programming*. Addison-Wesley Professional, 2002 (vid. pág. 37).

[40] BECK. *Extreme programming explained: embrace change.* Addison-Wesley Professional, 2000 (vid. pág. 37).

[41] Ana del Carmen Espinosa Robert. «Sistema Auditoría de Datos: Módulo de Gráficos v2.0». Tesis doct. Universidad de las Ciencias Informáticas, 2015 (vid. págs. 38, 61).

[42] J. J. Gutiérrez y col. «PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA». En: *Department de Lenguajes y Sistemas Informáticos University of Sevilla* (2006) (vid. pág. 39).

[43] Universidad de las Ciencias Informáticas. *Centro de Entornos Interactivos 3D (VERTEX)*. Universidad de las Ciencias Informáticas. 2017. URL: https://www.uci.cu/investigacion%20y%20desarrollo/centros%20de%20desarrollo/centro%20de%20entornos%20interactivos%203d%20vertex (vid. pág. 56).

[44] Lester Collado Rolo. *Centro de Entornos Interactivos 3D Vertex*. Caimanes. 2018. URL: https://facultad4.uci.cu/centro%20de%20entornos%20interactivos%203d%20vertex%202/ (vid. pág. 56).

[45] Alicia Bárbara Expósito Santana. «Pruebas de caja blanca Prueba y Mantenimiento del Software». En: (2012) (vid. págs. 60, 61).

Generado con LaTeX: 15 de junio de 2018: 4:11pm

Apéndices

APÉNDICE A

Anexos

A.1. Centro de Entornos Interactivos 3D (VERTEX)

El Centro de Entornos Interactivos 3D (VERTEX) adscrito a la Facultad 4, desarrolla productos y servicios informáticos asociados a los Entornos Interactivos 3D con un alto valor agregado, resultado de un ciclo completo de Investigación + Desarrollo + innovación (I+D+i) [43]. Entre sus funciones se encuentra [44]:

- Participar activamente en la informatización del país desarrollando soluciones que requieran de la Interacción con Entornos Virtuales 3D.
- Consolidar los productos y servicios ofrecidos como rubros exportables, que cuenten con una alta competitividad y estén basados en principios de software libre y reusabilidad.
- Formar personal altamente calificado en la investigación y desarrollo de soluciones con un alto sentido de la profesionalidad, competitividad y compromiso en la transformación de la sociedad.
- Promover alianzas estratégicas con instituciones líderes en la temática.

A.2. Modelado de la Base de Datos de Medicando

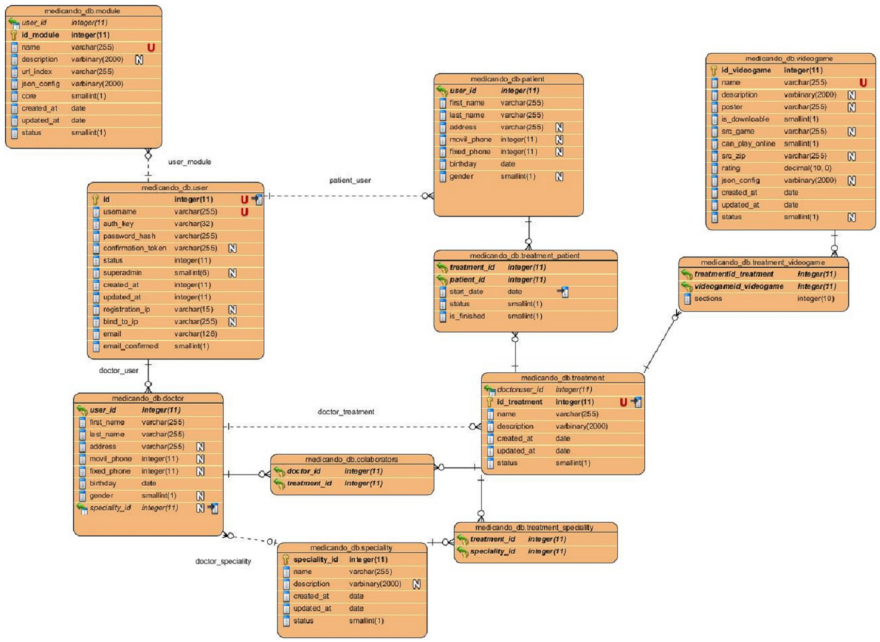


Figura A.1. Modelado de la Base de Datos de Medicando

A.3. Tareas de Ingeniería

Tareas de ingeniería para la I Iteración

- ***HU Administrar notificaciones***

Tabla A.1. Tarea de ingeniería # 10

Tarea	
Número de tarea: 10	**Número de Historia de usuario:** 1
Nombre de la tarea: Eliminación de notificaciones en estado leídas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.7
Fecha de inicio: 17 de noviembre de 2017	**Fecha de fin:** 1 de diciembre de 2017
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad, que permite eliminar un conjunto de notificaciones previamente seleccionadas que ya el usuario haya leído y no desea que vuelvan a ser listadas.	

- ***HU Enviar notificación***

Tabla A.2. Tarea de ingeniería # 11

Tarea	
Número de tarea: 11	**Número de Historia de usuario:** 2
Nombre de la tarea: Enviar notificaciones a los pacientes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.0
Fecha de inicio: 6 de enero de 2018	**Fecha de fin:** 20 de enero de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad que permite al doctor seleccionar un paciente en específico. Luego utilizando una vista, permite llenar los siguientes campos y enviar el mensaje al paciente especificado previamente: • Asunto (Obligatorio) • Mensaje (Obligatorio)	

Tareas de ingeniería para la II Iteración

- ***HU Mostrar notificación***

Tabla A.3. Tarea de ingeniería # 12

Tarea	
Número de tarea: 12	**Número de Historia de usuario:** 3
Nombre de la tarea: Mostrar todas las notificaciones	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 7 de febrero de 2018	**Fecha de fin:** 14 de febrero de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se implementa una funcionalidad que permita al usuario una vez que esté en su menú de notificaciones pendientes, pueda asceder a todas sus notificaciones, que están almacenadas en la base de datos.	

Tabla A.4. Tarea de ingeniería # 13

Tarea	
Número de tarea: 13	**Número de Historia de usuario:** 3
Nombre de la tarea: Marcar todas como leídas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 15 de febrero de 2018	**Fecha de fin:** 22 de febrero de 2018
Programador responsable: Daylilis Cardoso Roque	

Continúa en la próxima página

Tabla A.4. Continuación de la página anterior

Descripción: Se implementa una funcionalidad que permita al usuario una vez que esté en su menú de notificaciones pendientes, marcarlas todas como notificaciones leídas, cambiando el estado de las notificaciones en la base de datos.

Tareas de ingeniería para la Iteración III

- ***HU Administrar trazas***

Tabla A.5. Tarea de ingeniería # 14

Tarea	
Número de tarea: 14	**Número de Historia de usuario:** 5
Nombre de la tarea: Vistas para visualizar la información de la traza seleccionada	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 1 de mayo de 2018	**Fecha de fin:** 15 de mayo de 2018
Programador responsable: Daylilis Cardoso Roque	
Descripción: Se desarrolla una vista para mostrar la información recopilada sobre la traza en cuestión.	

A.4. Pruebas

A.4.1. Pruebas Unitarias

Método de prueba aplicado

Este método de caja blanca está centrado en la estructura de control del programa. Permite examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

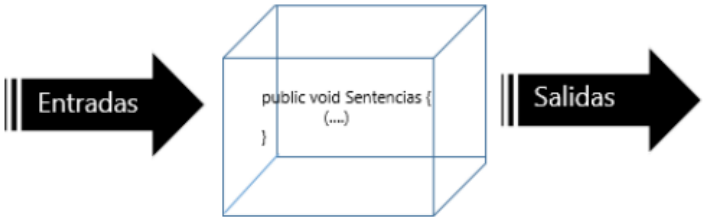


Figura A.2. Representación del método de caja blanca

Una de las técnicas más empleadas por este método para comprobar el funcionamiento correcto de las estructuras de los datos es la técnica del camino básico.

Camino Básico

Esta técnica permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta

medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez [45].

Para realizar la prueba es necesario realizar primeramente el análisis de la complejidad del algoritmo sobre el que se va a realizar la prueba, con el fin de calcular los valores de la complejidad ciclomática, como se muestra en la figura A.3.

```
public function actionDeleteSelected()
{
1  if (Yii::$app->request->isAjax) {
2      Yii::$app->response->format = Response::FORMAT_JSON;
       $ids = Yii::$app->request->post('selection');
3      if (isset($ids)) {
4          foreach ($ids as $id) {
5              $this->findModel($id)->delete();
           }
       }
   }
6  return "200 OK";
}
```

Figura A.3. Ejemplo del algoritmo Eliminar de notificaciones de manera simultánea

El grafo del flujo asociado al algoritmo Eliminar Tratamiento representa los caminos de ejecución que se pueden seguir en el procedimiento. Es un grafo orientado en el que los vértices representan instrucciones o conjuntos de instrucciones que se ejecutan como una unidad. Cada arista representa la posibilidad de que una vez terminada la ejecución de un vértice se pase a ejecutar otro.

Los componentes de dicho grafo son:

1. Los nodos que son círculos representados en el grafo de flujo que representan una o más secuencias del procedimiento, donde cada nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al principio y al final del grafo.
2. Los nodos predicados que son los que contienen una condición y se caracterizan porque de ellos salen una o más aristas.
3. Las aristas son las flechas del grafo, iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando este no represente la sentencia de un procedimiento.
4. Las regiones son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo como una región más. Las regiones se enumeran: siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico del procedimiento.

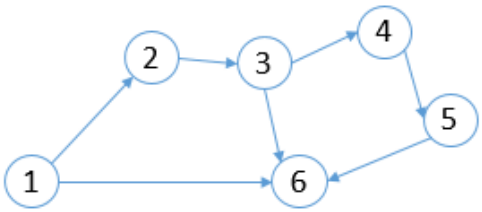


Figura A.4. Grafo de flujo asociado al algoritmo Eliminar de notificaciones de manera simultánea

La complejidad ciclomática es un parámetro de los grafos de flujo y representa la complejidad lógica de ejecución del programa. En el contexto de las pruebas, la cuantía de la complejidad ciclomática representa el número de caminos independientes que forman un conjunto de caminos básicos y por ello nos da el número mínimo de pruebas que garantiza la ejecución de cada instrucción al menos una vez [45].

La complejidad ciclomática se representa por *V(G)* y se puede calcular de varias formas:

- **V(G)= Aristas - Nodos + 2**= 7-6+2= 3
- **V(G) = Nodos predicados + 1**= 2 + 1 = 3
- Como el grafo tiene dos regiones, V(G) = 3

A.4.2. Pruebas de aceptación

Pruebas de Caja Negra

El método de caja negra tiene como objetivo fundamental la interacción con el software, entendiendo qué es lo que hace, sin dar importancia a cómo lo hace. Este método se enfoca a las necesidades del cliente, lo involucra como un miembro más del equipo, cumpliendo con las características que propone la metodología XP. Para su realización debe estar muy bien definida la interfaz del módulo. Con este tipo de prueba se intenta detectar funciones incorrectas o ausentes, errores de interfaz e incluso problemas de rendimiento [41].

Figura A.5. Representación del método de caja negra

Pruebas de aceptación para la Iteración I

Tabla A.6. Prueba de aceptación # 10

Caso de prueba de aceptación	
Código: HU1_P4	**Historia de usuario:** 1
Nombre: Eliminar notificaciones con estado leídas	
Descripción: Prueba para la funcionalidad administrar notificaciones (eliminar notificaciones leídas).	
Condiciones de ejecución: • El usuario debe estar previamente autenticado y contar con rol Administrador.	
Pasos de ejecución: 1. El administrador entra a la vista para la configuración del módulo Notificaciones. 2. Se marca la opción Eliminar notificaciones leídas. 3. Se eliminan de la base de datos las notificaciones a nivel de sistema con estado leídas. Luego se muestra un mensaje indicando que se han eliminado correctamente las notificaciones.	
Resultados esperados: Satisfactorio	

Tabla A.7. Prueba de aceptación # 11

Caso de prueba de aceptación	
Código: HU2_P6	**Historia de usuario:** 2
Nombre: Enviar notificación a paciente (caso positivo)	
Descripción: Prueba para la funcionalidad enviar notificación al paciente por parte de un doctor.	
Condiciones de ejecución: • El usuario deberá estar autenticado y contar con el rol de doctor. • El doctor autenticado debe tener al menos un paciente asignado. • El paciente a notificar debe tener habilitado la opción de recibir este tipo de notificación en su configuración personal.	
Pasos de ejecución: 1. El doctor autenticado, lista los pacientes que tiene asignados. 2. Se selecciona el primer paciente listado. 3. Se marca la opción Notificar y como respuesta del sistema, se muestra un formulario con los campos a llenar. 4. Se llenan los campos: *Asunto y Mensaje*. 5. Se selecciona la opción *Enviar*. 6. Se retorna a la vista de los pacientes del doctor.	
Resultados esperados: Satisfactorio	

Tabla A.8. Prueba de aceptación # 12

Caso de prueba de aceptación	
Código: HU2_P7	**Historia de usuario:** 2
Nombre: Enviar notificación a paciente (caso negativo)	
Descripción: Prueba para la funcionalidad enviar notificación al paciente por parte de un doctor.	
Condiciones de ejecución: • El usuario deberá estar autenticado y contar con el rol de doctor. • El doctor autenticado debe tener al menos un paciente asignado. • El paciente a notificar debe tener habilitado la opción de recibir este tipo de notificación en su configuración personal.	
Pasos de ejecución: 1. El doctor autenticado, lista los pacientes que tiene asignados. 2. Se selecciona el primer paciente listado. 3. Se marca la opción Notificar y como respuesta del sistema, se muestra un formulario con los campos a llenar. 4. Se llena solo el campo: *Mensaje*. 5. Se selecciona la opción *Enviar*. 6. Se muestra un mensaje de error indicando que existen campos vacíos, por lo que no se envía el mensaje. correctamente.	
Resultados esperados: Satisfactorio	

Pruebas de aceptación para la Iteración II

Tabla A.9. Prueba de aceptación # 13

Caso de prueba de aceptación	
Código: HU3_P3	**Historia de usuario:** 3
Nombre: Mostrar todas	
Descripción: Prueba para la funcionalidad mostrar todas.	
Condiciones de ejecución: • El usuario deberá estar autenticado. • El usuario debe tener al menos una notificación en el sistema.	

Continúa en la próxima página

Tabla A.9. Continuación de la página anterior

Pasos de ejecución: 1. El usuario desde la vista inicial de el sistema, preciona el equino de notificaciones pendientes(campana) que aparece en la esquina superior del banner. 2. Como respuesta, el sistema; muestra en una lista desplegable todas las notificaciones pendientes que dicho usuario tenga en el sistema, además da la posibilidad de seleccionar las funcionalidades:*Ver todas y Marcar todas como leídas*. 3. El usuario selecciona la opción *Ver Todas* 4. Como respuesta, el sistema; redirecciona a una vista en la cual se muestran en forma de tabla todas las notificaciones que dicho usuario tenga en el sistema, como la opción de:*ver y eliminar* cada una de ellas.
Resultados esperados: Satisfactorio

Tabla A.10. Prueba de aceptación # 14

Caso de prueba de aceptación	
Código: HU3_P4	**Historia de usuario:** 3
Nombre: Marcar todas como leídas	
Descripción: Prueba para la funcionalidad marcar todas como leídas.	
Condiciones de ejecución: • El usuario deberá estar autenticado. • El usuario debe tener al menos una notificación pendiente por visualizar.	
Pasos de ejecución: 1. El usuario marca el menú de notificaciones pendientes. 2. Como respuesta, el sistema; muestra en una lista desplegable todas las notificaciones pendientes que dicho usuario tenga en el sistema, además da la posibilidad de seleccionar las funcionalidades:*Ver todas y Marcar todas como leídas*. 3. El usuario selecciona la opción *Marcar todas como leídas* 4. Como respuesta, el sistema; elimina dichas notificaciones de las lista de notificaciones pendientes.	
Resultados esperados: Satisfactorio	