Universidad de las Ciencias Informáticas

Facultad 2

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

Gestión de avisos del Sistema para el control Farmacológico

Autores:

Gerardo Javier Acosta González

Juan Carlos Prado López

Tutores:

Ing. Yoiler Joaquin Frometa Moreno

Ing. Dannier Flores Ramos

La Habana

Junio de 2013

"Año 56 de la Revolución"

Concentra todos tus pensamientos en el trabajo que estás haciendo.

Los rayos de sol no queman hasta que se concentran en un punto.

Alexander Graham Bell (1847-1922)

.

Datos de Contacto

Ing. Dannier Flores Ramos (dflores@uci.cu): graduado de Ingeniero en Ciencias Informáticas en la UCI. Pertenece al Centro de Informática Médica (CESIM). Se desempeña como Jefe del Proyecto Sistemas para el control Farmacológico en el Departamento de Sistemas Especializados en Salud.

Ing. Yoiler Joaquin Frometa Moreno (yfrometa@uci.cu): graduado de Ingeniero en Ciencias Informáticas en la UCI. Pertenece al Centro de Informática Médica (CESIM). Se desempeña como Desarrollador del proyecto Sistemas para el control Farmacológico en el Departamento de Sistemas Especializados en Salud.

Agradecimientos

De Gerardo

A mis padres, a mis abuelos y a toda mi familia en general por ayudarme siempre en todo momento brindándome su apoyo.

A mi hermosa novia Gleidys, por comprenderme, dar apoyo y confianza, por darme cariño cuando más lo necesito y a mi suegra Nieves.

A mis tutores, Yoiler y Dannier por haberme guiado a como desempeñar una buena investigación y convertirme en un buen ingeniero.

A las profesoras Annia Arencibia y Aimara Marin, que a pesar de no ser las tutores directas, siempre estuvieron dispuestas a ayudarme ante cualquier duda que tuviera.

A mis compañeros de aula que desde los primer año hemos estado juntos luchando por graduarnos y que hemos estudiado juntos siempre apoyandonos unos a los otros.

A mi compañero de tesis, el Juanca que a pesar de su carácter y su locura, aportó su grano de arena para lograr este trabajo.

A mi Toshiba, por estar ahí conmigo para arriba y para abajo, aguantando más de quince horas encidada todos los días para lograr este trabajo.

Agradecimientos

De Juan Carlos

A Dios en primer lugar por hacer mi sueño realidad, por darme fe para creer que con mis conocimientos y la ayuda de mis compañeros podía llegar a graduarme en la Universidad más prestigiosa de Cuba.

A mi familia que siempre me apoyó y confió en mí.

A mis tutores porque no hubo ningún problema que le planteáramos al que no le buscaran una solución y a mi compañero de tesis por haberme acompañado en esta batalla de la cual hemos salido victoriosos.

Resumen

En el Sistema para el control Farmacológico (Synta), existe información que los usuarios deben conocer de manera automática para la toma decisiones. En dicho sistema se necesita conocer esta información sin necesidad de que los usuarios tengan que realizar búsquedas manuales o generar reportes. Con el fin de cumplir estas necesidades, se desarrollaron funcionalidades que generan avisos en el sistema Synta.

Los avisos están asociados al módulo en que se encuentre trabajando el usuario, el nivel de la entidad a la que se subordine y el rol. Algunos avisos serán configurables de manera que la gestión sea flexible al usuario final. Para el desarrollo de la solución, se utilizó Symfony 1.4 como marco de trabajo de desarrollo y Doctrine 1.2 como librería de Mapeo Objeto-Relacional.

Las funcionalidades de generación de avisos permitirán conocer de manera automática información relevante para la toma de decisiones. Con los avisos se podrán conocer la cantidad de reacciones adversas que esté provocando un medicamento, facilitando la monitorización de los mismos. Se conocerán además los usuarios próximos a caducar, los profesionales que hayan sobrepasado el plan de entrega de recetas así como los pacientes que lleven más de seis meses de tratamiento, consumiendo un medicamento controlado, entre otros.

Palabras clave: aviso, evento, información, toma de decisión

Índice

ÍNDICE

Introducción

Las TIC (Tecnologías de la Información y las Comunicaciones) son el conjunto de tecnologías que permiten el acceso, producción, tratamiento y comunicación de información presentada en diferentes códigos como son: texto, imagen y sonido. Giran en torno a tres medios básicos: la informática, la microelectrónica y las telecomunicaciones; pero giran, no sólo de forma aislada, sino lo que es más significativo, de manera interactiva e interconectadas, lo que permite conseguir nuevas realidades comunicativas. (Ortí, 2012)

La incorporación de las TIC en el mundo de la medicina tiene un papel primordial, favoreciendo el desarrollo de herramientas dirigidas a informatizar los procesos en áreas como la planificación, la investigación, la gestión, la prevención, promoción o en el diagnóstico o tratamiento. El reto actual es conseguir que las posibilidades que las TIC ponen a disposición de la sociedad, contribuyan a una mejora de la calidad de vida y el bienestar de las personas y ayuden a disminuir los desequilibrios y las desigualdades del acceso a los servicios de salud a los ciudadanos, optimización de la relación coste beneficio, a la vez que favorecen su desarrollo y crecimiento. En definitiva unos sistemas de salud más integrados y no sólo más interconectados. (González, 2007)

En la actualidad los sistemas informáticos desempeñan un papel importante en la solución de problemas específicos existentes en la sociedad, automatizando ciertas tareas como pueden ser la contabilidad, la gestión de una empresa, entre otras. Estos tienen como objetivo fundamental satisfacer las necesidades del usuario, ahorrándole tiempo y dinero, y facilitándole su trabajo.

En Cuba se viene desarrollando desde 1997 la informatización del Sistema Nacional de Salud (SNS), que no es más que la aplicación gradual e integral de las nuevas tecnologías de la información y las comunicaciones en la dirección de los procesos relacionados con las funciones del SNS a los diferentes niveles de atención. Ya desde hace años se han estado produciendo y poniendo en práctica soluciones informáticas para la informatización de algunos procesos administrativos, docentes, investigativos y asistenciales. (MINSAP, 2013)

En la Universidad de las Ciencias Informáticas (UCI) existen varios centros de desarrollo de software, entre los que se encuentra, el Centro de Informática Médica (CESIM). Dicho centro tiene la misión de

desarrollar productos, servicios y soluciones informáticas para el sector de la salud, contribuyendo a la formación integral de profesionales y permitiendo un posicionamiento en el mercado nacional e internacional. (UCI, 2013) Entre las aplicaciones informáticas que se han desarrollado en este centro, y que han contribuido en el proceso de informatización del SNS, se encuentran, el Sistema de Ingeniería Clínica y Electromedicina (SIGICEM), el Sistema de Gestión de Ensayos Clínicos (SIGEC) y la Red Cubana de Nefrología.

Actualmente en el CESIM, se desarrolla el Sistema para el control Farmacológico (Synta), ejemplo de solución informática vinculada al campo de la medicina. Concretamente esta es una aplicación web que gestiona información relacionada con la distribución de las recetas médicas, las inscripciones al consumo de medicamentos controlados, el consumo de medicamentos de forma general y las reacciones adversas provocadas por los mismos. Esta gestión se realiza a través de los módulos Control de Recetas Médicas, Tarjeta Control, Consumo de Medicamentos y Reacciones Adversas a Medicamentos (RAM).

El Módulo Control de Recetas Médicas se encarga de la gestión de la distribución de recetas médicas en el SNS cubano, debido a que las instituciones de Salud Pública no poseen un control estricto de las recetas que le fueron entregadas y las recetas que la propia unidad de salud ha entregado, ya sea a otra unidad o simplemente a un profesional, siendo este el último eslabón de la cadena de entrega de recetas médicas. Este sistema estandariza los procesos de entrega, recibos e incidencias en la entrega de recetas médicas, constituyendo estas, la herramienta que asegura ante la emisión de un diagnóstico por parte de un facultativo, la indicación consecuente de un medicamento para el tratamiento de la patología diagnosticada. (Morales, 2012)

El Módulo Tarjeta Control realiza la gestión de toda la información referente a la vigilancia de la prescripción de medicamentos en el Departamento de la Farmacoepidemiología del Ministerio de Salud Pública (MINSAP). El MINSAP tiene un Programa Nacional de Medicamentos el cual contempla una lista de medicamentos que se controlan mediante una tarjeta de control. Dicha tarjeta contiene la indicación de los medicamentos que serán consumidos por los pacientes; estos son fármacos controlados debido al riesgo que implica su uso en cuanto a la dependencia a la que quedan expuestos los consumidores. Estos medicamentos sumaron un total de 84 dentro del Sistema de Salud Pública cubano en el 2011, aunque la cifra puede variar de un año a otro. (Menéndez, 2012)

El Módulo Consumo de Medicamentos gestiona la información referente al consumo de medicamentos en el MINSAP. Permite realizar varias acciones como calcular distintos indicadores que se utilizan para conocer la disponibilidad y utilización de los fármacos, generar distintos reportes de evolución, modificación o distribución en el Cuadro Básico de Medicamentos (CBM), entre otras. Con este módulo se puede tener una medida directa o indirecta de hechos relevantes de una sociedad dada como la morbilidad y hábitos de consumo de una población. (Lorenzo, y otros, 2011)

El Módulo Reacciones Adversas a Medicamentos es el encargado de gestionar todas las reacciones adversas a los fármacos registradas en la nación. Dicho módulo permite realizar el análisis del comportamiento de las reacciones a través de reportes y gráficas.

En los módulos Control de Recetas Médicas, Tarjeta Control, Reacciones Adversas a Medicamentos y Administración existe información que al alcanzar determinados valores o comportamientos deben ser analizados por el personal especializado para la toma de decisiones. A continuación se describen algunos ejemplos de relevancia:

- Módulo Control de Recetas Médicas: el registro de incidencias en el proceso de distribución de las recetas es una de las informaciones más importantes gestionadas por el módulo, sin embargo cuando se registra alguna de estas, el sistema no es capaz de informar su ocurrencia. Esto implica que no se tomen las medidas administrativas en tiempo, lo que puede provocar pérdidas económicas por irregularidades en el proceso o actos delictivos como el robo de las recetas.

- Módulo Tarjeta Control: la cantidad de población por provincia es un dato necesario para el cálculo de indicadores como la tasa de inscripción de medicamentos por provincia o nación, este puede cambiar anualmente por lo que es necesario su actualización. El módulo no es capaz de informar cuándo este valor está incompleto, lo que puede ocasionar que los cálculos obtenidos no sean reales y la toma de decisiones errónea.

- Módulo Tarjeta Control: gestiona las inscripciones de pacientes que consumen medicamentos controlados, pero no permite realizar una búsqueda detallada de los pacientes que llevan más de seis meses de tratamiento. Estos pacientes son analizados mensualmente para ver si han consumido o no el medicamento, eliminando la inscripción en caso negativo, de esta forma se

evitan gastos al Estado Cubano debido a que se realiza una planificación más real de las necesidades de la población. Actualmente se hace difícil este proceso en las farmacias, por lo que solo se analizan las inscripciones de los medicamentos más necesitados, provocando pérdidas económicas y que personas que realmente los necesitan no tengan acceso a los mismos.

- Módulo RAM: se registra las reacciones adversas a medicamentos que ocurran en la nación, sin embargo no es posible obtener de manera automática la cantidad de reacciones adversas que esté provocando un medicamento, conocer rápidamente esta singularidad permitiría al personal correspondiente tomar medidas y así evitar que un medicamento pudiese seguir afectando a un grupo poblacional.

- Módulo Administración: no se conocen las cuentas de usuarios próximas a caducar. Esta información es importante que los administradores la conozcan de manera oportuna, para que tomen medidas al respecto pues si un usuario se le vence la cuenta, no podrá interactuar con el sistema.

Actualmente en el sistema Synta, para saber si existe alguno de estos valores o comportamientos alarmantes, es necesario realizar búsquedas, generar reportes o gráficas para visualizar y analizar la información, ya que este no cuenta con un mecanismo que informe de la ocurrencia de algún evento.

A partir de la problemática planteada se define como **problema a resolver:** ¿Cómo informar a los usuarios del Sistema para el control Farmacológico, ante la ocurrencia de eventos o comportamientos importantes, contribuyendo a una mayor inmediatez en la toma de decisiones?, por lo que se toma como **objeto de estudio:** el proceso de generación de avisos en aplicaciones web.

Según lo planteado anteriormente se establece como **campo de acción:** el proceso de generación de avisos en el Sistema para el control Farmacológico.

Quedando definido de esta manera como **objetivo general:** desarrollar una solución de gestión de avisos en el Sistema para el control Farmacológico.

Para darle cumplimiento al objetivo general se definieron las siguientes **tareas de investigación:**

1. Analizar los sistemas de avisos existentes a nivel nacional e internacional estableciendo similitudes con la investigación en curso.
2. Asimilar las herramientas, tecnologías y metodología, definidas para el desarrollo del Sistema para el control Farmacológico.
3. Obtener los artefactos requeridos correspondientes a los flujos de trabajo indicados por la metodología de desarrollo Proceso Unificado de Desarrollo, sirviendo de guía para la realización de la solución.
4. Implementar la solución propuesta en el Sistema para el control Farmacológico alcanzando la gestión de avisos acorde a las necesidades del cliente.

Para la realización de las tareas se utilizan métodos de investigación en la búsqueda y procesamiento de la información. Estos métodos se dividen en teóricos y empíricos.

Métodos Teóricos

- **Histórico-Lógico:** se utilizó en la búsqueda de los antecedentes del objeto de la investigación, al hacer una exploración de aplicaciones que contengan módulos o componentes que generen avisos y de soluciones previas existentes a problemas similares al actual. También permitió identificar posibles mejoras y alternativas de solución para la generación de avisos.
- **Analítico-Sintético:** se empleó para el análisis a partir de fuentes bibliográficas confiables, de conceptos y técnicas empleadas en soluciones previamente desarrolladas. Permitió además descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.
- **Modelación:** se aplicó para representar mediante diagramas y modelos, el proceso de gestión de avisos, facilitando el entendimiento de la solución desarrollada.
- **Sistémico:** se utilizó para describir los conceptos asociados a la gestión de avisos y la relación entre ellos como elementos fundamentales a tener en la investigación.

Métodos Empíricos

- **Observación:** permitió conocer la realidad mediante la percepción directa de los objetos y fenómenos relacionados con la generación de avisos en aplicaciones web.
- **Entrevista:** se entrevistó a la analista del Sistema para el control Farmacológico, con el objetivo de comprender el flujo de negocio que se lleva a cabo en dicho sistema y entender de manera más explícita, las funcionalidades que se quieren alcanzar con la solución de la presente investigación. La entrevista fue no estructurada y la misma se encuentra el Anexo 1 de este documento.

Una vez desarrollada la solución de gestión de avisos del Sistema para el control Farmacológico se obtendrán los siguientes beneficios:

- Los usuarios recibirán información primordial relacionada con el trabajo que realizan mediante mensajes de avisos, facilitándoles la toma de decisiones al obtener la información de manera automática.
- Permitirá a los usuarios configurar determinados avisos en dependencia de sus necesidades siendo así una gestión de avisos flexible a cambios.
- El Sistema para el control Farmacológico se fortalecerá como software ya que contará con nuevas funcionalidades.

El documento cuenta con la siguiente estructura:

Capítulo 1: Fundamentación Teórica

Se realiza un estudio del estado del arte sobre los sistemas de avisos utilizados en las aplicaciones web en el mundo y en Cuba. Se describe la metodología, las tecnologías y herramientas a utilizar para el desarrollo de la solución informática.

Capítulo 2: Características del Sistema

Se describe la propuesta de solución, el modelo de dominio, los requisitos funcionales y no funcionales con que contará la solución de gestión de avisos en el sistema para el control farmacológico. También se representa el diagrama de casos de uso del sistema y las descripciones de los casos de uso, así como se detallan los actores del sistema.

Capítulo 3: Diseño del Sistema

Ofrece una representación de la realización de los casos de uso del sistema mediante el diagrama de clases del diseño y las descripciones de estos. Asimismo se definen los patrones de diseño y de arquitectura a utilizar.

Capítulo 4: Implementación

Se detallan aspectos de interés para el proceso de implementación como el modelo de datos y descripción de los estándares y estilos de codificación a utilizar. Igualmente se plantea el diagrama de despliegue y el diagrama de componentes.

Capítulo 1: Fundamentación Teórica

Introducción

En el capítulo se profundiza la manera de contextualizar el problema de investigación anteriormente planteado mediante un estudio del estado del arte. El estado del arte es un compendio escrito de artículos, libros y otros documentos que describen el estado pasado y actual del conocimiento sobre el problema de estudio. Ayuda a documentar y agregar valor a la investigación así como sustentarla teóricamente. (Hernández Sampieri, y otros, 2006)

1.1 Vocablos asociados a la investigación

Aviso:

Se puede definir un aviso como noticia o advertencia que se comunica a alguien. Significa además atención, una señal o una advertencia. (RAE, 2001)

Notificar:

Dar con propósito cierto, noticia de algo. (RAE, 2001)

Evento:

Un evento de manera general se puede definir como: suceso o hecho considerado de interés o importancia. (VOX, 2011) Específicamente para el campo de la informática un evento se denomina de la siguiente forma: una acción que es detectada por un programa; éste, a su vez, puede hacer uso del mismo o ignorarlo. (Definición de, 2014)

1.2 Sistemas de avisos o de notificaciones

Las notificaciones en el campo de la informática hoy en día, se emplean en diversos dispositivos tales como: computadoras personales, teléfonos móviles y consolas de videojuegos para mantener a los usuarios al tanto de una serie diversa de cuestiones. La noción de notificación está estrechamente vinculada con un aviso. El envío de una notificación tiene como objetivo, dejar presentada determinada información, que es de vital importancia para una persona tener conocimiento de la misma. En el ámbito de la informática, las notificaciones aparecen relacionadas con alertas que emiten ciertos programas o

servicios, para advertir o actualizar a un determinado usuario interesado en algo específico. (Definicion de, 2014)

Es muy común la integración de sistemas de avisos o de notificaciones en las aplicaciones web, ejemplo de ello es la red social más popular del mundo, Facebook, la cual permite mostrar notificaciones a los usuarios en la misma interfaz web o enviarlas por correo electrónico o teléfono celular. Las técnicas de notificación utilizadas en aplicaciones web, permiten a los usuarios recibir notificaciones tan pronto como la información es publicada, sin necesidad de chequear la fuente original manualmente para obtener actualizaciones. (Werdmuller, 2010)

Las notificaciones han evolucionado cada vez más en los últimos años, debido al auge de la utilización de las tecnologías de la información y las comunicaciones en conjunto con el aumento de las redes y la necesidad de establecer una permanente comunicación entre los usuarios y las empresas. Todo con el objetivo de mantener informado y actualizado a la persona interesada en un tema específico, constituyendo así una estrecha relación usuario – empresa.

Las notificaciones resultan de gran importancia en diferentes contextos de la sociedad, entre los cuales se encuentra el comercio, ya que mediantes éstas, los propietarios de un determinado producto o servicio informan promociones sobre las ofertas que tienen. También para la prensa las notificaciones han tenido una gran aceptación, por medio de éstas, los periódicos digitales pueden mantener actualizado de las diferentes noticias y acontecimientos a sus suscriptores. Igualmente en el campo de la medicina, las notificaciones o avisos son de importancia, pues permiten informar por ejemplo, a un paciente cuando estén listos los resultados de los exámenes que le fueron realizados o de avisar a un médico cuando tiene planificada una cirugía.

1.2.1 Elementos que componen un sistema de notificaciones

Un sistema de notificaciones genérico está formado por los siguientes elementos: (Pérez Santiesteban, y otros, 2013)

- Manejador de eventos: encargado de la detección u ocurrencia de sucesos o eventos en el sistema.
- Manejador de notificaciones: organiza las acciones a realizarse después de detectado un evento, busca la relación con los usuarios existentes y arma el mensaje o notificación a enviarse.

- Base de datos: almacena la información para que se generen los eventos, los datos de los usuarios y los mensajes a notificar.
- Dispositivos: destino de la información, dispositivos tales como: número de teléfono celular, dirección de correo electrónico, direcciones IP y nombres de dominio.
- Sistema de envío: ejecuta el envío de acuerdo al dispositivo registrado por el usuario en el sistema, por lo general el manejador de notificaciones es el encardado de ejecutar acciones de envío.

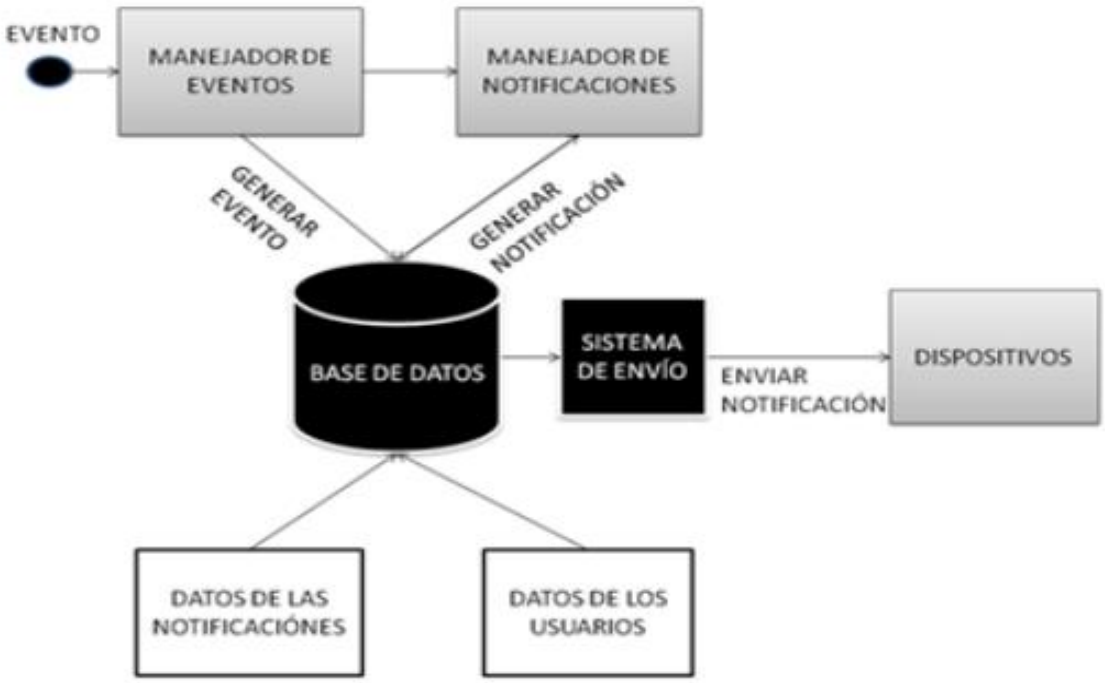


Figura 1. Elementos que componen un sistema de notificaciones genérico. (Pérez Santiesteban, y otros, 2013)

A continuación se analizan y describen varios sistemas que presentan diferentes maneras de gestionar notificaciones. Los análisis de estos sistemas servirán de base para la obtención de experiencias tanto positivas como negativas, las cuales se tendrán en cuenta para el desarrollo de la solución del problema a resolver en la presente investigación.

1.2.2 Sistemas existentes a nivel internacional

1.2.2.1 Facebook

Considerada una de las redes sociales más importantes y populares de la actualidad, Facebook es una interfaz virtual desarrollada en el año 2004 por cuatro estadounidenses de la ciudad de Cambridge, Massachusetts: Mark Zuckerberg, Dustin Moskovitz, Eduardo Saverin y Chris Hughes. Una de las características más importantes de Facebook es su constante actualización y es debido a esto que uno puede, en su perfil, observar instantáneamente las diferentes acciones, decisiones y actividades que realizan los contactos de su lista. (Definición ABC, 2013) Todas estas actualizaciones en tiempo real se logran en gran parte gracias a la gestión de notificaciones existentes en Facebook.

Esta red social en su apartado de configuración permite dos opciones: “Cómo recibes notificaciones” y “Notificaciones que recibes”. La primera permite configurar la manera en que el usuario recibe las notificaciones, eligiendo entre, la misma interfaz web de Facebook, recibirlas en el correo electrónico o teléfono celular. En la segunda opción se puede configurar, qué tipo específico de notificaciones y de qué amigos se desea recibir.

1.2.2.2 Blackboard Learn

Blackboard Learn es una compañía norteamericana que provee una plataforma de programas de aprendizaje en línea que contiene diversos cursos para la educación superior, corporaciones y organizaciones gubernamentales. Actualmente sus programas y servicios son muy utilizados en las universidades de los Estados Unidos de América y en otros países del mundo.

Blackboard Learn cuenta con un sistema para la entrega de notificaciones a sus usuarios cuando se produce un evento determinado, como la creación de una actividad, el envío de una encuesta o el vencimiento del plazo de una prueba. Dependiendo de los parámetros definidos por los administradores, los usuarios individuales podrán configurar qué notificaciones desean recibir y seleccionar cómo quieren recibir sus notificaciones. (Blackboard Learn, 2013)

Las notificaciones se pueden enviar a los destinatarios de dos formas: (Blackboard Learn, 2013)

- En línea: Pueden aparecer en las páginas de inicio del curso o en un panel de notificaciones, este panel es una página que muestra notificaciones de usuario en todos los cursos.

- Distribuciones: Las notificaciones pueden enviarse a través de uno o más mecanismos de distribución. En la mayoría de los casos, se envían por correo electrónico.

Todas las notificaciones se generan automáticamente siempre que se produce el evento asociado a las mismas. Además las notificaciones que aparecen en el panel, cuentan con enlaces a las páginas que contienen información relacionada con el evento que desencadena la notificación, para que el usuario pueda consultar o realizar una acción. También el sistema establece un nivel de prioridades en las notificaciones y las procesa según su prioridad.

1.2.3 Sistemas existentes a nivel nacional

1.2.3.1 Sistema de Información Hospitalaria

Como parte del Sistema de Información Hospitalaria (por sus siglas en inglés HIS) se diseñó un componente para el envío de notificaciones y alertas. Su objetivo es hacer llegar información de interés personal y administrativo a pacientes y trabajadores dentro y fuera de una institución hospitalaria en el menor tiempo posible. Por ejemplo, notificar a los médicos cuando tienen planificada consultas con los pacientes o avisar a los responsables de un banco de sangre cuando las bolsas estén próximas a vencerse.

La gestión de notificaciones del sistema utiliza tres canales distintos de comunicación para suministrar las notificaciones, estos son, vía mensajes de texto (SMS, por sus siglas en inglés) de telefonía celular, correo electrónico y beeper en tiempo real. Para el diseño del componente se utilizó la metodología de desarrollo de software Proceso Unificado de Desarrollo, la cual se apoyó en el Lenguaje de Modelado Unificado y en la Notación para el Modelado de Procesos del Negocio (BPMN, por sus siglas en inglés). Además, para la creación del prototipo no funcional del sistema fue utilizado el entorno de desarrollo Eclipse, unido a Java como lenguaje de programación y PostgreSQL como Sistema Gestor de Base de Datos. (Estanque Díaz, y otros, 2013)

1.2.3.2 Sistema de Gestión Universitaria

El Sistema de Gestión Universitaria (SGU), es un sistema informático desarrollado en la Universidad de las Ciencias Informáticas con el objetivo de alinear los procesos sustantivos de la universidad. Gestiona una amplia gama de información mediante varios subsistemas como son: Pregrado, Postgrado, Extensión Universitaria, Investigaciones, Producción, Residencia entre otros.

En estos subsistemas ocurren importantes eventos como la baja de estudiantes en la universidad, entre otros, que no pueden pasar inadvertidos y necesitan ser informados a los distintos directivos de la universidad para la toma de decisiones. Es por esto que el SGU cuenta con un módulo de notificaciones y alertas como solución para un rápido acceso a la información generada por dichos eventos previamente mencionados.

Este módulo de notificaciones y alertas implementa funciones que se comportan como manejadores de notificaciones y manejadores de alertas, los cuales se encargan de organizar las acciones a realizarse después de la ocurrencia de un evento o el análisis de una variable; en ambos casos se arma el mensaje o notificación a enviarse, se buscan las relaciones con los usuarios y se ejecuta el envío de acuerdo al dispositivo registrado. Los dispositivos que maneja como destino de la información son: dirección de correo electrónico, buzón de notificaciones y alertas del SGU. Fue desarrollado con CodeIgniter 1.7.2 como marco de trabajo, PostgreSQL 8.4.2 como gestor de base de datos, Scrum y Programación Extrema como metodología de desarrollo, Lenguaje Unificado de Construcción de Modelos 2.0 como lenguaje de modelado, entre otras tecnologías. (Pérez Santiesteban, y otros, 2013)

1.3 Resultados de la investigación

Los sistemas Facebook y Blackboard Learn son de tipo software propietario, lo que impide acceder a su código fuente para adaptarlo a otras necesidades. La causa fundamental por la que se descartarán la gestión de notificaciones de los sistemas internacionales anteriormente descritos, es debido a que incumplen con una de las principales políticas para el desarrollo de la informática en el sector de la salud en Cuba. Dicha política plantea que todos los productos y servicios se integrarán a la ciber-infraestructura del sector y se realizarán en lo fundamental sobre sistemas abiertos (…) utilizando software libre y de calidad. (Rodríguez, y otros, 2007)

Sin embargo, en la gestión de notificaciones de Facebook, el modo de configurar las notificaciones es muy simple y con una interfaz intuitiva al usuario, característica que se tendrá en cuenta a la hora de realizar la gestión de avisos de la actual investigación. De igual forma, la gestión de notificaciones de Blackboard Learn, aporta características positivas a la investigación, pues emplea un panel específico para mostrarlas, aspecto importante a destacar, porque mantiene organizada todas las notificaciones en un solo lugar y además con hipervínculo a la página que contiene la información relacionada con el evento disparador de la misma.

El mecanismo de notificaciones del Sistema de Información Hospitalaria difiere en cuanto a las tecnologías usadas en su desarrollo respecto a las utilizadas en Synta, ya que este último recurre al lenguaje de programación PHP y gestor de base de datos MySQL. Tampoco sería factible reutilizar este mecanismo, puesto que en el sistema Synta se pretende mostrar los avisos en la interfaz web y no enviar avisos por correo electrónico, beeper o mensajes de celular.

La gestión de notificaciones del SGU se encuentra integrado al negocio de dicho sistema, y utiliza el marco de trabajo de desarrollo, CodeIgniter, distinto al utilizado en el desarrollo de Synta, este último tiene definido Symfony como marco de trabajo de desarrollo. Además el SGU, envía las notificaciones por correo electrónico y hacia un buzón de notificaciones, mientras que Synta, requiere mostrar los avisos en la misma interfaz web y no enviarlos a un buzón de notificaciones o por correo electrónico. Debido a estas razones no sería factible reutilizar este módulo del SGU. No obstante aporta experiencias a la investigación, como la característica de tener un manejador de notificaciones para organizar las acciones a ejecutar una vez ocurrido un evento.

1.4 Metodología, tecnologías y herramientas a utilizar

La metodología, tecnologías y herramientas a utilizar en el desarrollo de la solución fueron definidas por el departamento de Sistemas Especializados en Salud. En el documento SAS - Synta - Arquitectura de software.doc que se encuentra en el expediente de proyecto, se define que para el desarrollo del sistema Synta se utilizará como lenguaje de programación PHP 5.3, marco de trabajo de desarrollo Symfony 1.4, librería de JavaScript Ext JS 3.3 y gestor de base de datos MySQL 5.0. A continuación se profundizará en las características de estas así como del lenguaje de modelado UML 2.1 y de la metodología de desarrollo Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés).

1.4.1 Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo es una metodología de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. RUP es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones y de tamaño de proyecto. Está basado en componentes interconectados a través de interfaces bien definidas. RUP utiliza Lenguaje Unificado de Modelado para preparar todos los esquemas de un sistema software. (Jacobson, y otros, 2000). Además permite documentar todo el proceso de desarrollo.

Posee tres aspectos claves que lo define: (Jacobson, y otros, 2000)

- Dirigido por casos de uso: un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Todos los casos de uso en su conjunto constituyen el modelo de casos de uso el cual describe la funcionalidad completa del sistema. Los casos de usos se modelan y representan los requisitos funcionales y guían el proceso de desarrollo de software, ya que los modelos que se obtienen a través de los flujos de trabajo, representan la realización de los mismos.

- Centrado en la arquitectura: la arquitectura es la estructura general que contempla las partes más relevantes de un sistema lo que permite tener una visión común entre los desarrolladores y los usuarios, indica cómo debe ser construido el sistema y en qué orden. Al definirse la arquitectura se toma en cuenta los elementos de calidad, reutilización y capacidad de evolución del sistema. Los casos de uso y la arquitectura están estrechamente relacionados, por una parte los casos de uso deben acoplar a la arquitectura que se lleve a cabo y esta debe permitir el desarrollo de los casos de uso que se requieran tanto en el sistema que se desarrolle en curso como en un futuro.

- Iterativo e incremental: el proceso iterativo e incremental de RUP propone dividir el desarrollo de software en partes más pequeñas denominadas iteración, esta comprende actividades en todos los flujos de trabajo fundamentales con la cual se obtiene un incremento del sistema que se está desarrollando.

1.4.2 Lenguaje de Modelado UML 2.1

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. (Rumbaugh, y otros, 2007)

1.4.3 Enterprise Architect versión 7.1

Enterprise Architect (EA) es una herramienta de Ingeniería de Software Asistida por Computadoras (CASE, por sus siglas en inglés) de uso muy sencillo, que aborda el diseño y análisis UML y cubre el desarrollo de software desde la captura de requerimientos a lo largo de las etapas de análisis, diseño, pruebas y mantenimiento. EA es una herramienta multiusuario, diseñada para ayudar a construir software robusto y fácil de mantener. Además, permite generar documentación e informes flexibles y de alta calidad. (Sparx Systems, 2008)

Algunas de sus principales características son: (Sparx Systems, 2008)

- Construido sobre la base de UML 2.1.
- Velocidad, estabilidad y rendimiento.
- Trazabilidad completa, desde el análisis de requerimientos y los artefactos de diseño, hasta la implementación y el despliegue.

Esta herramienta CASE permite crear a los desarrolladores de una manera fácil, los distintos diagramas y modelos necesarios para el desarrollo de la solución informática de la presente investigación. Además, EA integra una guía de ayuda bien documentada, donde explica detalladamente todas las funcionalidades que se pueden realizar con la herramienta.

1.4.4 PHP 5.3

PHP es un lenguaje de programación interpretado de alto nivel y de código abierto, especialmente pensado para el desarrollo de aplicaciones web interpretadas y ejecutadas del lado del servidor, además puede ser incrustado en páginas HTML. PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo, Solaris y OpenBSD), Microsoft Windows y Mac OS X. Soporta la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros. Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos (como Oracle, PostgreSQL y MySQL). (Achour, y otros, 2013)

En la actualidad, PHP se encuentra en segundo lugar entre los diez lenguajes más usados en el mundo (ver Anexo 2), cuenta con una gran comunidad de desarrolladores y una amplia documentación en línea por lo que resulta fácil encontrar información referente a este lenguaje. También presenta gran rapidez en la ejecución y bajos requerimientos de consumo en los sistemas donde es desplegado.

1.4.5 JavaScript 1.10

JavaScript es un lenguaje de programación basado en objetos y que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como aparición y desaparición de texto, animaciones, acciones que se activan al pulsar botones u otros elementos y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (Pérez, 2009)

1.4.6 Symfony 1.4

Es un marco de trabajo que simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener, facilita la programación de sistemas, ya que encapsula operaciones complejas en instrucciones sencillas. (Potencier, y otros, 2010)

Su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia MIT de software libre. La documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos. (Symfony.es, 2014)

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura Modelo Vista Controlador que permite separar la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (Potencier, y otros, 2010)

Symfony es un marco de trabajo de desarrollo fácil de instalar y configurar, ya sea en el sistema operativo Windows o Linux. Symfony propone una estructura jerarquizada de directorios que permite tener una mayor organización en la solución que se está desarrollando, esto facilita el mantenimiento y las ampliaciones futuras de la aplicación. También presenta la característica de que es independiente del sistema gestor de bases de datos que se esté usando. Además es fácil de extender, pues permite integración con librerías desarrolladas por terceros. Para una mayor comprensión, en el Anexo 3 del presente trabajo se encuentra el diagrama de flujo de trabajo de Symfony.

1.4.7 AJAX

JavaScript asíncrono y XML (AJAX, por sus siglas en inglés), no es un lenguaje de programación, sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que permite hacer páginas de internet más interactivas. (Ajax Ya, 2014) AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor. (Eguiluz Pérez, 2013)

1.4.8 Ext JS 3.3

Es una librería JavaScript de código abierto de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Entre sus ventajas está el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente. (Frederick, et al., 2008) Sus características principales son, gran desempeño, componentes de interfaz de usuario personalizables, buen diseño y documentación. (Desarrollo Web, 2014) Presenta una licencia comercial, es decir, esta se debe comprar cuando se desarrolla un software propietario, y una licencia pública general (GPL, por sus siglas en inglés), que se aplica cuando se desarrolla un proyecto de código abierto, lo que implica liberar el mismo con licencia GNU GPL V3.

Ext JS 3.3 posee entre sus características, buen rendimiento en aplicaciones web complejas, al igual que provee formularios con diseños de óptima calidad e intuitivos al usuario. Fue utilizado en el desarrollo de todos los módulos de Synta, conviniendo una correcta integración entre este último y la solución de gestión de avisos a desarrollar en la presente investigación.

1.4.9 Doctrine 1.2

Doctrine es una librería de Mapeo Objeto-Relacional (ORM, por sus siglas en inglés), situado encima de una capa de abstracción de bases de datos (DBAL, por sus siglas en inglés). La principal tarea de este ORM es traducir de forma trasparente los objetos PHP en filas de una base de datos relacional y viceversa. Una de las características claves que tiene Doctrine es escribir consultas en un dialecto propio similar al SQL, llamado Doctrine Query Language (DQL), inspirado en HQL de Hibernates. Doctrine tiene una gran comunidad de desarrolladores y se integra con los principales marcos de trabajo de desarrollo web como son Symfony, Zend y CodeIgniter. (Doctrine Project, 2014) De manera que es muy sencillo, gracias a este ORM, cambiar de un sistema de bases de datos a otro completamente diferente en cualquier etapa del desarrollo de un proyecto, permitiendo así una mayor flexibilidad en la solución que se desarrolla.

1.4.10 Servidor Web Apache 2.2

Apache es un servidor HTTP, de código abierto y licenciamiento libre, que funciona en Linux, sistemas operativos derivados de Unix, y Windows. Ha desempeñado un papel muy importante en el crecimiento de la red mundial, y continua siendo el servidor HTTP más utilizado, siendo además el servidor de facto contra el cual se realizan las pruebas comparativas y de desempeño para otros productos competidores. Apache es desarrollado y mantenido por una comunidad de desarrolladores auspiciada por la Fundación Apache. (Apache.org, 2014)

Tiene como características que es flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos HTTP, modular, es decir, puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos. Es extensible, gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP. (Open Suse.org, 2014)

1.4.11 MySQL 5.0

El software MySQL proporciona un servidor de base de datos SQL muy rápido, multihilo, multiusuario y robusto. El servidor MySQL está diseñado para entornos de producción con alta carga de trabajo así como para integrarse en software para ser distribuido. Tiene una doble licencia, una de código abierto por lo que puede usarse de forma gratuita bajo licencia GPL y otra comercial estándar. (MySQL, 2014)

Tiene como principales características que permite recurrir a bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a diferentes necesidades y requerimientos, desarrolla alta velocidad en la búsqueda de datos e información. (Definición ABC, 2014) El servidor está disponible como un programa separado para usar en un entorno de red cliente-servidor, además cuenta con un sistema de reserva de memoria muy rápido basado en hilos, tolera grandes bases de datos y soporte completo para distintos conjuntos de caracteres. (MySQL, 2014)

1.4.12 PhpMyAdmin 3.4

PhpMyAdmin es una herramienta que permite administrar bases de datos MySQL empleando un navegador web, tanto para administrarla local como remotamente. Permite crear o eliminar bases de

datos; crear, eliminar o alterar tablas, ejecutar consultas SLQ entre muchas más funciones. (phpMyAdmin, 2014)

Tiene como principales características: (phpMyAdmin, 2014)

- Software libre y multiplataforma.
- Interfaz web intuitiva.
- Posee amplia documentación en idioma Inglés y Español.
- Soporte de la mayoría de las características de MySQL.
- Administra usuarios y privilegios de MySQL.
- Importa datos desde archivos SQL.

1.4.13 NetBeans 7.3

NetBeans IDE 7.3 es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris que permite de manera rápida y fácil desarrollar aplicaciones Java para móviles, para escritorio y aplicaciones web. Es gratuito y de código abierto, cuenta con una comunidad de desarrolladores vibrante y ofrece extensa documentación y recursos de capacitación. También proporciona un gran conjunto de herramientas para PHP, JavaScript, Ajax, Ruby, Grails y C/C++ así como una variada selección de plugins de terceros. (Netbeans, 2014)

NetBeans posee integración con Symfony permitiendo desarrollar aplicaciones de forma más sencilla y productiva. En primer lugar, es posible crear nuevos proyectos y aplicaciones directamente desde el IDE. También se pueden ejecutar todas las tareas de Symfony, incluso pasándole argumentos y opciones, visualizando el resultado sin necesidad de utilizar una consola de comandos externa. Además, al editar el archivo de una vista se tiene acceso al autocompletado de variables, incluso de los objetos del núcleo de Symfony y también permite saltar de una acción al archivo de su vista asociada y viceversa. (Symfony.es, 2009)

Capítulo 2: Características del Sistema

Introducción

En el capítulo se describe la propuesta de solución para la gestión de avisos del Sistema para el control Farmacológico. Para ello se detalla el modelo de dominio, se declaran los requisitos funcionales y no funcionales así como se identifican los actores del sistema. Además se presentan los diagramas de casos de uso del sistema con sus descripciones, favoreciendo un mejor entendimiento para los desarrolladores de la solución informática.

2.1 Modelo de dominio

Un modelo de dominio describe los conceptos importantes del contexto de un sistema como objetos del dominio, y enlaza estos objetos unos con los otros. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en que trabaja el sistema. (Jacobson, y otros, 2000)

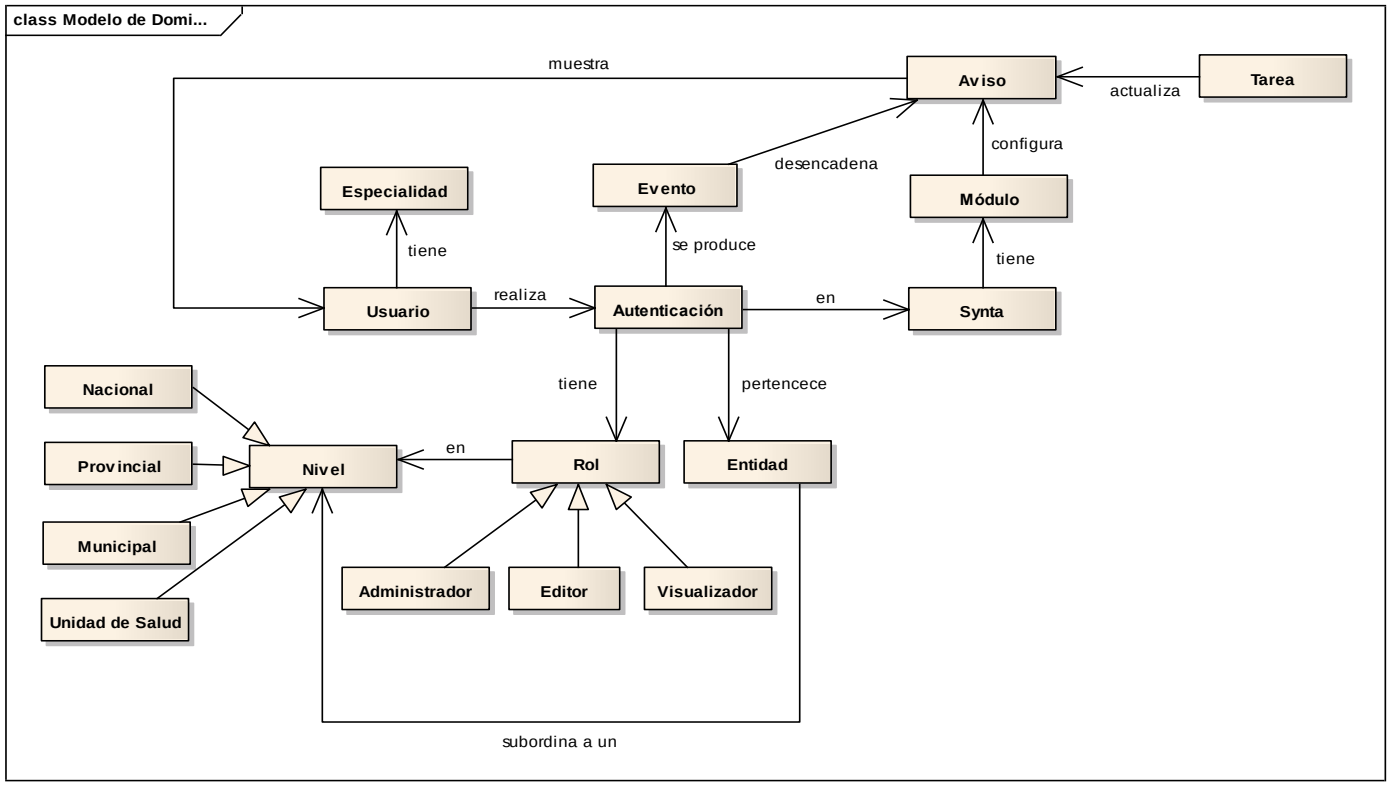


Figura 2. Modelo de dominio (Fuente: Realización propia)

2.1.1 Definición de los conceptos del modelo de dominio

- Usuario:

Personal que tiene acceso a los diferentes módulos que forman parte del Sistema para el control Farmacológico.

- Especialidad:

Representa la especialidad del profesional de la salud: Distribuidor de Recetas, Farmacéutico o Farmacoepimediólogo.

- Autenticación:

Es el proceso de detectar y comprobar la identidad del usuario examinando las credenciales del mismo, validando esas credenciales contra las existentes en el módulo de Administración.

- Evento:

Representa el acto de autenticación de un usuario en el Sistema para el control Farmacológico y que desencadena la generación de un aviso.

- Aviso:

Constituye la noticia o advertencia que se genera y se le muestra al usuario en el Sistema para el control Farmacológico.

- Rol:

Representa los roles que puede tomar un usuario: Visualizador, Editor o Administrador.

- Entidad:

Representa las instituciones de salud del MINSAP, estas se subordinan a un nivel.

- Nivel:

Representa los distintos niveles de dirección administrativa del SNS cubano, siendo estos Nacional, Provincial, Municipal o Unidad de Salud.

- Módulo:

Representa los distintos módulos que tiene el sistema y que contarán con avisos, estos son: Control de Recetas Médicas, Tarjeta Control, Reacciones Adversa a Medicamentos y Administración.

- Tarea:

Actualiza los avisos cada un tiempo determinado.

2.2 Propuesta de solución

Para dar respuesta a la situación problemática plasmada en el inicio de la presente investigación, se plantea desarrollar una solución informática para gestionar avisos en el Sistema para el control Farmacológico. Esta deberá garantizar que los usuarios reciban información relevante mediante avisos ante la existencia de determinados valores o comportamientos, facilitando la toma de decisiones. Los avisos se mostrarán automáticamente cuando el usuario realice el evento de autenticación en el sistema y seleccione el módulo en que va a trabajar.

Todos se presentarán en la interfaz de inicio de cada módulo en forma de mensajes, los cuales se clasificarán de dos tipos, en alertas y avisos, siendo estas las prioridades que tendrán los mismos. La clasificación alertas será para los mensajes que sean de mayor prioridad y se identificarán por un icono de color rojo, mientras que la clasificación avisos, será para los mensajes de prioridad normal y se identificarán con un icono de color azul, siendo la clasificación por defecto para todos los mensajes. Esto permitirá mostrar los mensajes ordenados atendiendo al orden de mayor prioridad.

Los avisos estarán asociados a los usuarios en dependencia del módulo, el nivel (nacional, provincial, municipal o unidad de salud) de subordinación de la entidad y al rol (visualizador, editor o administrador) que tengan establecidos en el módulo de Administración. De esta manera se pretende mantener la confidencialidad de la información que se maneja y que la misma esté disponible de manera oportuna, de tal forma que los usuarios puedan realizar con mayor facilidad la toma de decisiones. También se agregarán funcionalidades al sistema, que servirán de apoyo a los avisos que se reciban. Para ello, los avisos tendrán hipervínculo a interfaces con formularios que mostrarán los datos relacionados con el mismo, permitiéndole al usuario realizar acciones.

Además, los avisos se actualizarán cada quince minutos mientras que el usuario mantenga abierta la sesión del módulo en el que se encuentre. Conjuntamente, algunos que dependan de valores que puedan

variar, podrán ser configurables, es decir, los usuarios podrán predefinir ciertos parámetros de manera que ajusten los avisos a las necesidades que tengan en un momento determinado. Esto permite que la solución sea flexible al usuario final.

Basado en los elementos que componen un sistema de notificaciones genérico plasmado en el capítulo anterior, adaptado a las necesidades de la presente investigación y con el objetivo de mostrar un mayor entendimiento de la solución propuesta, se elabora el siguiente diagrama:

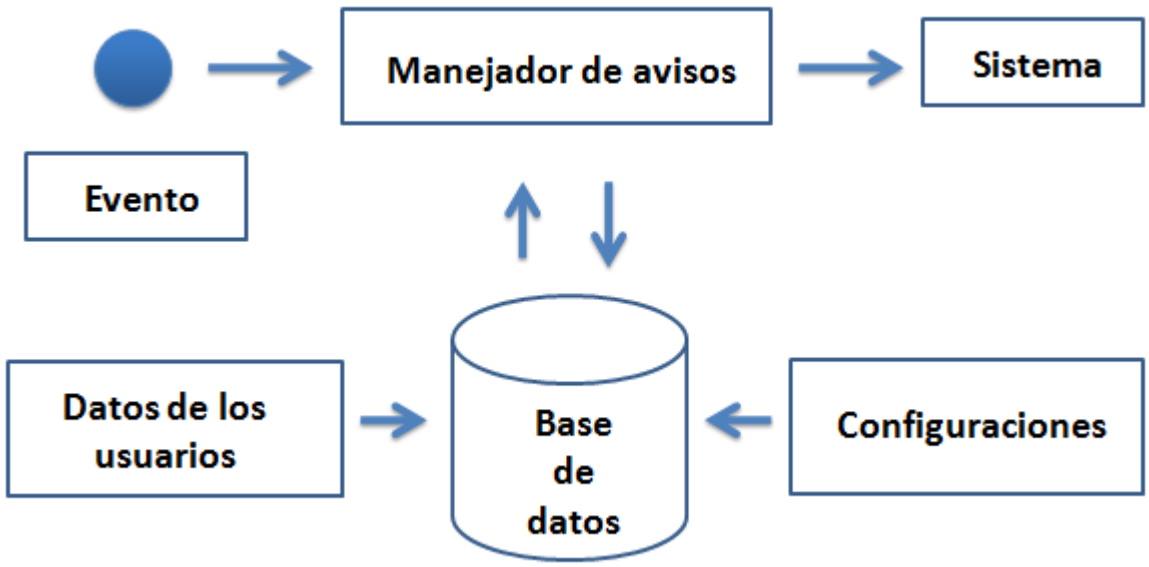


Figura 3. Diagrama de los elementos que componen la gestión de avisos propuesta (Fuente: Realización propia).

El evento representa la acción de autenticación de un usuario en el sistema. El manejador de avisos organiza las acciones a realizarse después de detectado el evento, busca en la base de datos la relación de los usuarios existentes, con las configuraciones que existen para un aviso determinado. Luego el manejador de avisos arma el mensaje y lo envía a la interfaz del sistema.

2.3 Requisitos de software

Una vez planteada la propuesta de solución, se procede a realizar la definición de los requisitos de software. La ingeniería de requisitos proporciona el mecanismo apropiado para entender lo que el software debe hacer. Permite analizar las necesidades, evaluar la factibilidad, especificar la solución sin

ambigüedades, validar la especificación, y administrar los requisitos conforme éstos se transformen en un sistema operacional. (Pressman, 2005)

2.3.1 Requisitos funcionales

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. 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. (Olivera, y otros, 2010) A continuación se listan los requerimientos funcionales (RF) para la realización de la gestión de avisos en el Sistema para el control Farmacológico.

Requisitos asociados al módulo Control de Recetas Médicas:

- RF 1 Listar avisos ante la ocurrencia de incidencias reportadas en las entregas de recetas médicas realizadas al nivel inferior.
- RF 2 Mostrar aviso ante la existencia de recibos sin confirmar.
- RF 3 Listar avisos con los nombres de los profesionales, entidades, municipios o provincias que sobrepasaron el plan de entregas en dependencia del nivel.
- RF 4 Configurar aviso de sobrepaso de plan de entregas.

Requisitos asociados al módulo Tarjeta Control:

- RF 5 Listar avisos ante la existencia de un registro de población incompleta.
- RF 6 Mostrar aviso de existencia de pacientes con más de seis meses de tratamiento.
- RF 7 Buscar pacientes inscritos por medicamentos por un período de un mes.
- RF 8 Buscar pacientes inscritos por medicamentos por un período de tres meses.
- RF 9 Buscar pacientes inscritos por medicamentos por un período de seis meses.
- RF 10 Eliminar inscripción de paciente asociado a un medicamento controlado.

Requisitos asociados al módulo Reacciones Adversa a Medicamentos:

- RF 11 Mostrar aviso ante la existencia de planillas del profesional o paciente sin revisar por el especialista.
- RF 12 Listar avisos sobre la cantidad de reacciones adversas provocadas por medicamentos.
- RF 13 Configurar aviso de cantidad de reacciones adversas que esté provocando un medicamento.

Requisitos asociados al módulo Administración:

- RF 14 Configurar aviso para mostrar al usuario el tiempo restante en que caducará la cuenta.
- RF 15 Configurar aviso para mostrar al administrador los usuarios próximos a caducar.
- RF 16 Listar avisos al administrador con los nombres de los usuarios próximos a caducar.

Requisito asociado a todos los módulos:

- RF 17 Mostrar aviso al usuario de caducidad de la cuenta.

2.3.2 Requisitos no funcionales

Un requisito no funcional (RNF) especifica los criterios que se deben usar para juzgar el funcionamiento de un sistema, en lugar de un comportamiento específico. Los requisitos no funcionales verifican cómo un sistema debería de ser, son a menudo llamados las “cualidades de un sistema”. (Softqanetwork, 2009). A continuación se describen los RNF para el desarrollo de la gestión de avisos del Sistema para el control Farmacológico.

RNF de Usabilidad:

- RNF 1 El sistema debe facilitar el uso para usuarios con pocos conocimientos en el campo de la informática.
- RNF 2 El sistema solo podrá ser utilizado por los usuarios definidos en el mismo.

RNF de Integridad:

- RNF 3 La información podrá ser modificada solo por personal autorizado.

RNF de Interfaz:

- RNF 4 La interfaz debe ser sencilla, no debe poseer muchas entradas que requiera gran entrenamiento para utilizar el sistema; agradable, ya que debe tener colores con tonalidades claras. El contenido será mostrado de manera comprensible y fácil de leer.
- RNF 5 Las interfaces de usuario deben estar definidas y creadas siguiendo las pautas establecidas por el departamento de Sistemas Especializados. (Ejemplo: íconos, mensajes del sistema, etiquetado).

RNF de Confidencialidad:

- RNF 6 La autenticación será la primera acción del usuario en el sistema y consistirá en suministrar un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica. Si el usuario autenticado no se encuentra registrado se debe reportar un error de acceso.
- RNF 7 Los usuarios podrán visualizar y configurar los avisos en dependencia del nivel de la entidad a la que se subordinen y los roles que tengan establecidos de acuerdo a la función que realizan.

RNF de Soporte:

- RNF 8 La implementación utilizará normas de codificación con el objetivo de obtener un producto comprensible y homogéneo; por lo que se utilizará la notación CamelCase para identificar las variables y las clases, y la notación PascalCase para los nombres de los métodos, lo cual hace el código más legible y facilita el soporte y mantenimiento de la aplicación.

RNF de Portabilidad:

- RNF 9 La solución informática será multiplataforma, es decir, se podrá ejecutar sobre sistemas operativos GNU/Linux y Windows.

RNF de Software:

Software para el cliente:

- RNF 10 Navegador web Mozilla Firefox versión 3.6, Internet Explorer versión 8.0 o versiones superiores.

- RNF 11 Sistema operativo Windows o GNU/Linux.

Software para el servidor:

Software servidor de base de datos:

- RNF 12 Utilizar gestor de base de datos MySQL 5.0.
- RNF 13 Sistema operativo GNU/Linux Debian 4.

Software servidor de aplicaciones:

- RNF 14 Utilizar servidor web Apache versión 2.2.
- RNF 15 Lenguaje PHP 5.3 y el marco de trabajo de desarrollo Symfony 1.4
- RNF 16 Sistema operativo GNU/Linux Debian 4.

RNF de Hardware:

Interfaces de hardware para el cliente:

- RNF 17 Computadora con 512 MB de memoria RAM, disco duro de al menos 1GB libre y tarjeta de red.

Interfaces recomendables de hardware para el servidor de base de datos:

- RNF 18 Computadora con microprocesador de velocidad 3.0 GHz, 3 GB de memoria RAM, disco duro de 80 GB y tarjeta de red.

Interfaces recomendables de hardware para el servidor de aplicaciones:

- RNF 19 Computadora con microprocesador de velocidad 3.0 GHz, 1 GB memoria RAM, disco duro de 10 GB y tarjeta de red.

2.4 Actores del sistema

Los actores del sistema son una agrupación uniforme de personas, sistemas o máquinas que interactúan con el sistema. Un actor es una clase de rol, mientras que un usuario es una persona que, cuando usa el sistema, asume un rol. (Ceria, 2013)

Tabla 1. Actores del sistema (Fuente: Realización propia)

Actor	Descripción
Usuario	Personal que tiene acceso a los diferentes módulos que forman parte del Sistema para el control Farmacológico. Tiene distintos roles según el módulo donde se autentique.
Distribuidor de Recetas	Recibirá solo los avisos que le corresponden según el nivel (nacional, provincial, municipal o unidad de salud) de subordinación de la entidad de salud a la que se subordine. Tendrá el privilegio de visualizar y modificar información relacionada con los avisos del módulo Control de Recetas Médicas.
Farmacéutico	Recibirá solo los avisos que le corresponden según el nivel (nacional, provincial, municipal o unidad de salud) de subordinación de la entidad de salud a la que se subordine. Tendrá el privilegio de visualizar y modificar información relacionada con los avisos de los módulos Tarjeta Control y Control de Recetas Médicas.
Farmacoepimediólogo	Recibirá solo los avisos que le corresponden según el nivel (nacional, provincial, municipal o unidad de salud) de subordinación de la entidad de salud a la que se subordine. Tendrá el privilegio de visualizar y modificar información relacionada con los avisos de los

	módulos Tarjeta Control y Reacciones Adversas a Medicamentos.
Administrador	Registra todos los datos y permisos de los usuarios del sistema. Tiene el privilegio de configurar los avisos del módulo de Administración.

2.5 Diagrama de casos de uso del sistema

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso. Sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen qué es lo que debe hacer el sistema, pero no cómo. (Hensgen, 2005)

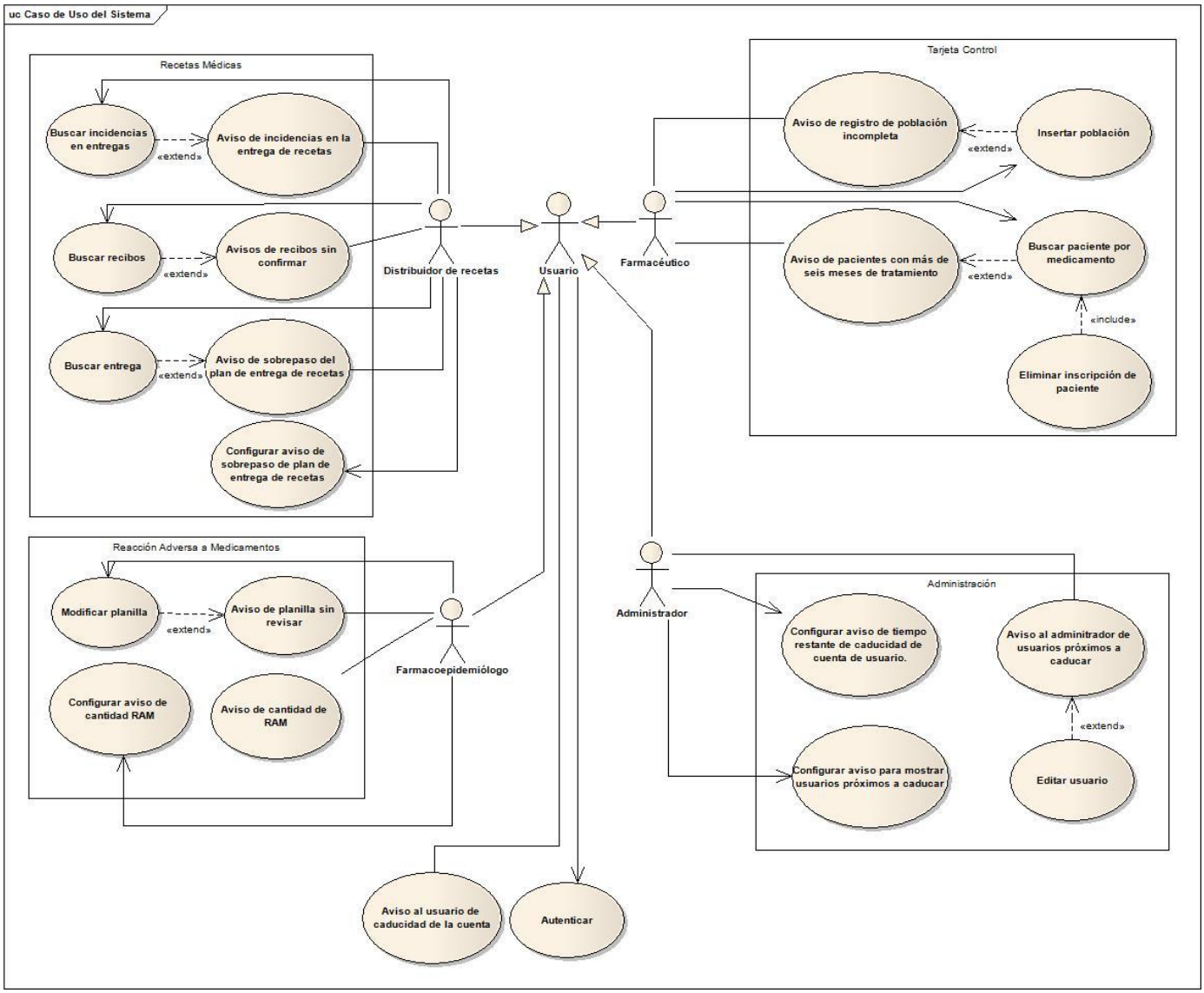


Figura 4. Diagrama de Casos de Uso del Sistema (Fuente: Realización propia)

2.6 Descripción de los casos de uso del sistema

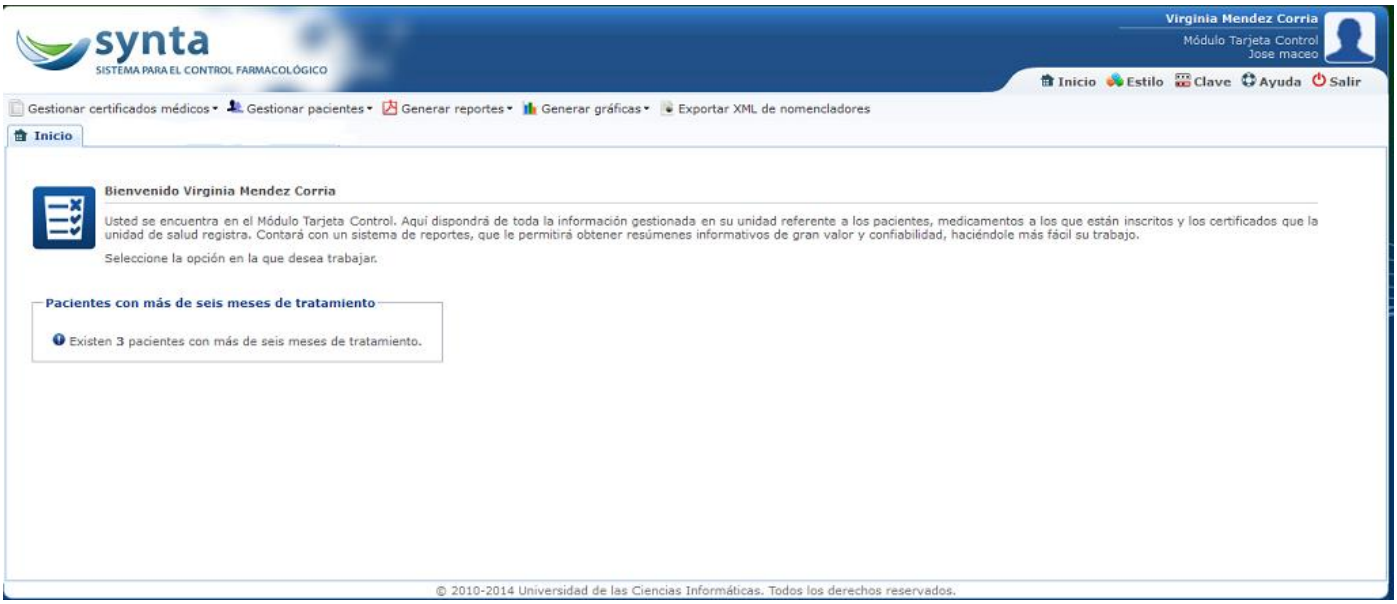
Seguidamente se describen algunos de los casos de usos para la gestión de avisos del Sistema para el control Farmacológico, el resto de las descripciones de los casos de uso de gestión de avisos se encuentran en el documento SES - Synta - Avisos_Especificación de casos de uso.

Tabla 2. CU Mostrar aviso de pacientes con más de seis meses de tratamiento

Objetivo	Mostrar un aviso cuando existan pacientes registrados con más de seis meses de tratamiento.
Actores	Farmacéutico
Resumen	El caso de uso inicia cuando el actor accede al módulo de Tarjeta Control. Seguidamente el sistema muestra en la interfaz de inicio del módulo, un aviso en forma de mensaje que indica la existencia de pacientes con más de seis meses de tratamiento.
Complejidad	Baja
Prioridad	Secundario
Precondiciones	Deben existir pacientes registrados en el sistema.
Postcondiciones	El aviso es mostrado.
Requisitos Funcional	RF 6
Flujo de eventos	
Flujo básico Mostrar aviso ante la existencia de pacientes con más de seis meses de tratamiento.	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el actor accede al módulo Tarjeta Control.	
2.		Muestra en la interfaz de inicio del módulo, un aviso en forma de mensaje que indica la existencia de pacientes con más de seis meses de tratamiento. Si el actor presiona el mensaje, ver descripción del caso de uso: Buscar pacientes con más de seis meses de tratamiento.

		En caso contrario ver flujos alternos.
3.		El caso de uso termina.
Flujos alternos		
Nº 1. No existen pacientes con más de seis meses de tratamiento		
	Actor	**Sistema**
1.		No muestra el aviso de pacientes con más de seis meses de tratamiento.
Prototipo no Funcional		



Relaciones	**CU Incluidos**	
	CU Extendidos	Buscar paciente por medicamento

Tabla 3. CU Aviso de incidencias en la entrega de recetas

Objetivo	Listar avisos cuando existan incidencias reportadas en las entregas

	de recetas médicas realizadas en el nivel inferior.
Actores	Distribuidor de recetas
Resumen	El caso de uso inicia cuando el actor accede al módulo Control de Recetas Médicas. Seguidamente el sistema muestra en la interfaz de inicio del módulo, un aviso con una lista de mensajes mostrando la cantidad de incidencias reportadas del nivel inferior respecto al nivel que tiene asignado dicho actor.
Complejidad	Baja
Prioridad	Secundario
Precondiciones	El usuario debe estar autenticado en el sistema.
Postcondiciones	El aviso es mostrado.
Requisitos Funcionales	RF 1

Flujo de eventos

Flujo básico Listar avisos de incidencias reportadas en las entregas de recetas en el nivel inferior.

	Actor	**Sistema**
1.	El caso de uso inicia cuando el actor accede al módulo Control de Recetas.	
2.		Muestra en la interfaz de inicio del módulo, un aviso con una lista de mensajes mostrando la cantidad de incidencias reportadas del nivel inferior respecto al nivel que tiene asignado dicho actor. Si el actor presiona el mensaje, ver descripción del caso de uso: Buscar incidencia en entrega En caso contrario ver flujo alterno 1.
3.		El caso de uso termina.

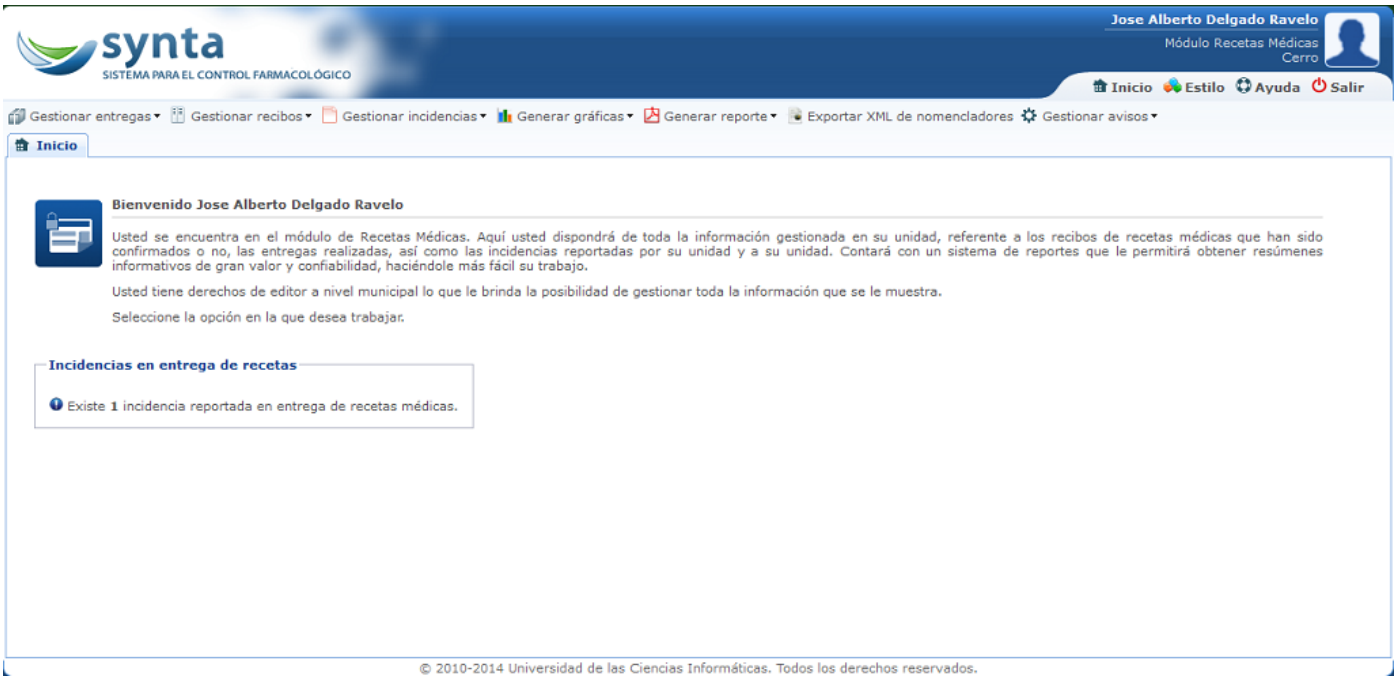
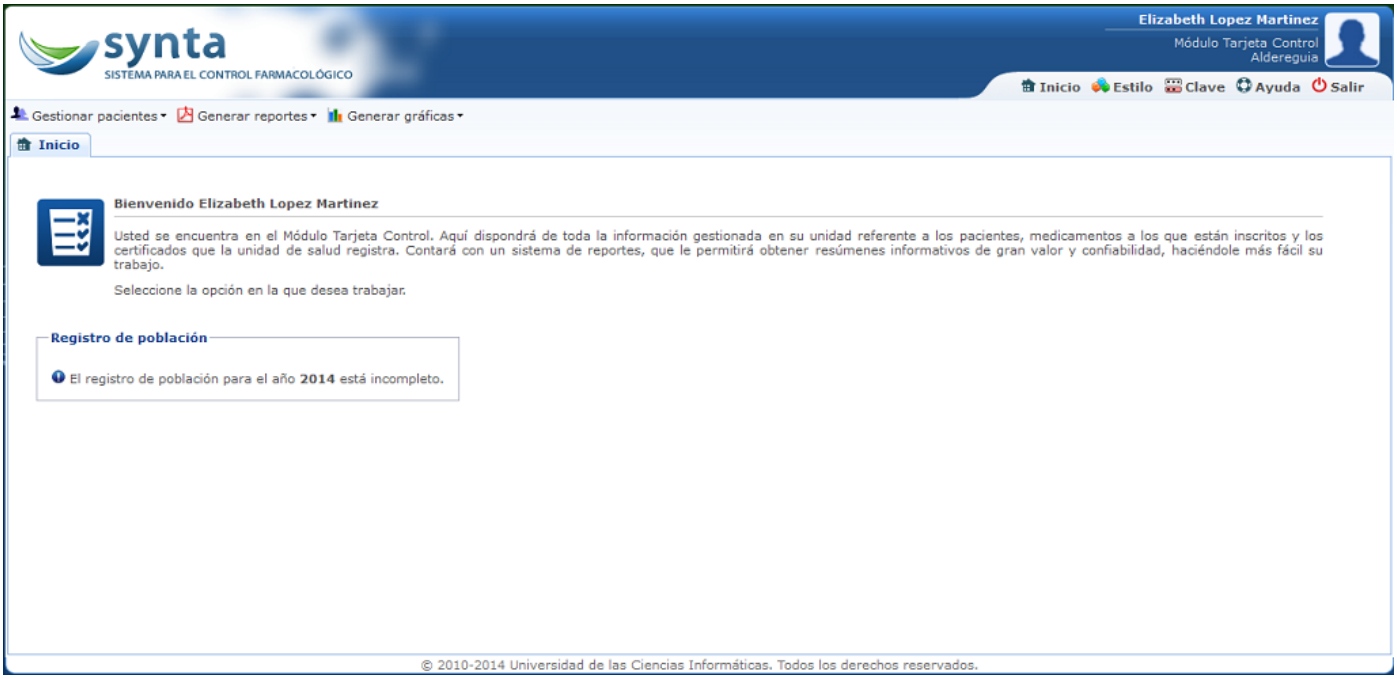
Flujos alternos		
Nº 1 No existen incidencias registradas en el sistema.		
	Actor	**Sistema**
		No muestra el aviso de incidencias en la entrega de recetas.
Prototipo no Funcional		
		
Relaciones	**CU Incluidos**	
	CU Extendidos	Buscar incidencia en entregas

Tabla 4. CU Aviso registro de población incompleta

Objetivo	Mostrar una lista de avisos ante la existencia de registros de población incompleta por años.
Actores	Farmacéutico

Resumen	El caso de uso inicia cuando el actor accede al módulo Tarjeta Control. Inmediatamente se muestra en la interfaz de inicio del módulo, un aviso con una lista de mensajes mostrando los registros de población incompleta por años.	
Complejidad	Baja	
Prioridad	Secundario	
Precondiciones	Debe existir algún registro de población.	
Postcondiciones	El aviso es mostrado.	
Requisitos Funcional	RF 5	
Flujo de eventos		
Flujo básico Mostrar aviso ante la existencia de registros de población incompleta		
	Actor	**Sistema**
1.	El caso de uso inicia cuando el actor accede al módulo Tarjeta Control.	
2.		Muestra en la interfaz de inicio del módulo, un aviso con una lista de mensajes mostrando los registros de población incompleta por años. Si el actor presiona el mensaje, ver descripción del caso de uso: Insertar población. En caso contrario ver flujo alterno 1.
3.		El caso de uso termina.
Flujos alternos		
Nº 1 No existe un registro de población incompleta.		

	Actor	Sistema
		No se muestra el aviso de registro de población incompleta.
Prototipo no Funcional		



Relaciones	**CU Incluidos**	
	CU Extendidos	Insertar población

Capítulo 3: Diseño del Sistema

Introducción

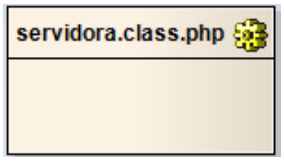
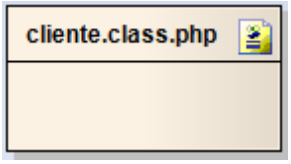
En este capítulo se transformarán los requerimientos definidos con anterioridad, a una propuesta de diseño, que será la guía a seguir para la implementación de la gestión de avisos. Además se realizará una breve descripción de los patrones de diseño y arquitectónicos a utilizar en la solución, así como se presentará el diagrama de clases del diseño con su respectiva descripción, para una comprensión de la solución a implementar a los desarrolladores.

3.1 Modelo de diseño

Es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación. (García Peñalvo, y otros, 2008)

3.1.1 Definición de los elementos del diseño

Tabla 5. Estereotipos web para las clases del diseño

Estereotipos web para las clases del diseño	
Estereotipo	**Descripción**
	Página Servidora: Representa una página web que tiene scripts que son ejecutados por el servidor. Estos scripts interactúan con recursos del servidor como bases de datos, lógica de negocio, sistemas externos y se encarga de construir (build) o generar el resultado HTML.
	Página Cliente: Es una página web con formato HTML en la cual se presentan los datos. Las páginas clientes son representadas por los navegadores clientes, y pueden contener scripts que son interpretados por el navegador.

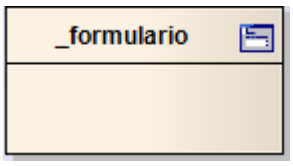
	Formulario: Es una colección de campos de entrada que forman parte de una página cliente. Los formularios envían sus datos al código servidor para ser procesados los pedidos (submit).

Tabla 6. Estereotipos para las relaciones entre las clases

Estereotipos para las relaciones entre las clases	
Estereotipo	**Descripción**
link	Representa un apuntador desde una “página cliente” hacia una “página cliente” o “página servidora”.
submit	Esta relación siempre ocurre entre un “formulario” y una “página servidora”, la “página servidora” procesa los datos que el “formulario” le envía.
build	Se establece cuando la “página servidora” crea una “página cliente”. Una “página servidora” puede crear varias “páginas clientes”, pero una “página cliente” sólo puede ser creada por una sola “página servidora”.
redirect	Es una relación unidireccional que indica que una página web redirecciona el procesamiento a otra página.

3.1.2 Diagrama de clases del diseño

Con el objetivo de facilitar la interpretación y mostrar las diferentes clases que componen la solución informática de la presente investigación, y cómo se relacionan unas con otras, se realizó el diagrama de clases del diseño. Para una mayor organización, cada elemento del diseño se encuentra en la capa que le pertenece teniendo en cuenta el patrón Modelo-Vista-Controlador.

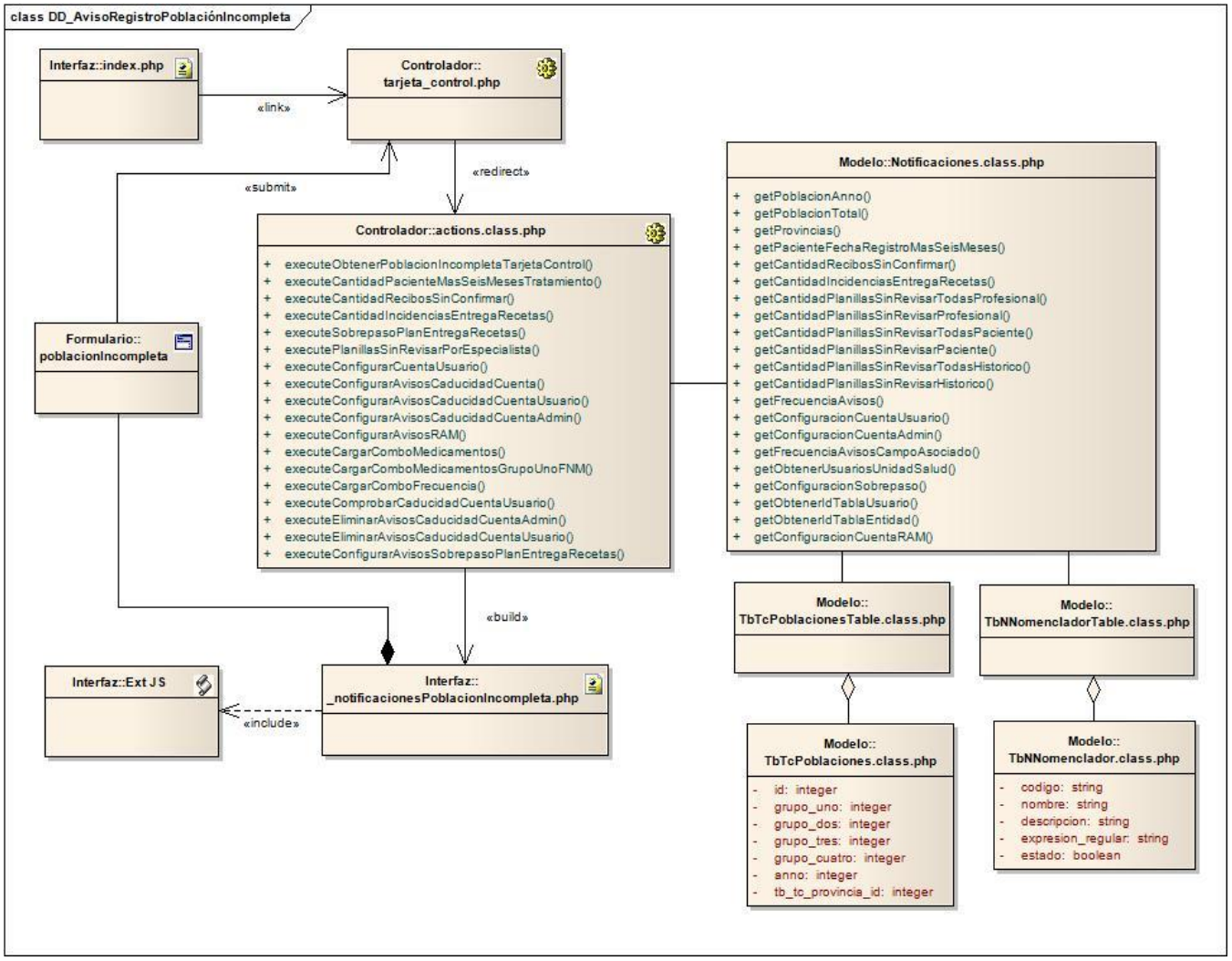


Figura 5. Diagrama de clase del diseño. Caso de uso: Aviso de registro de población incompleta

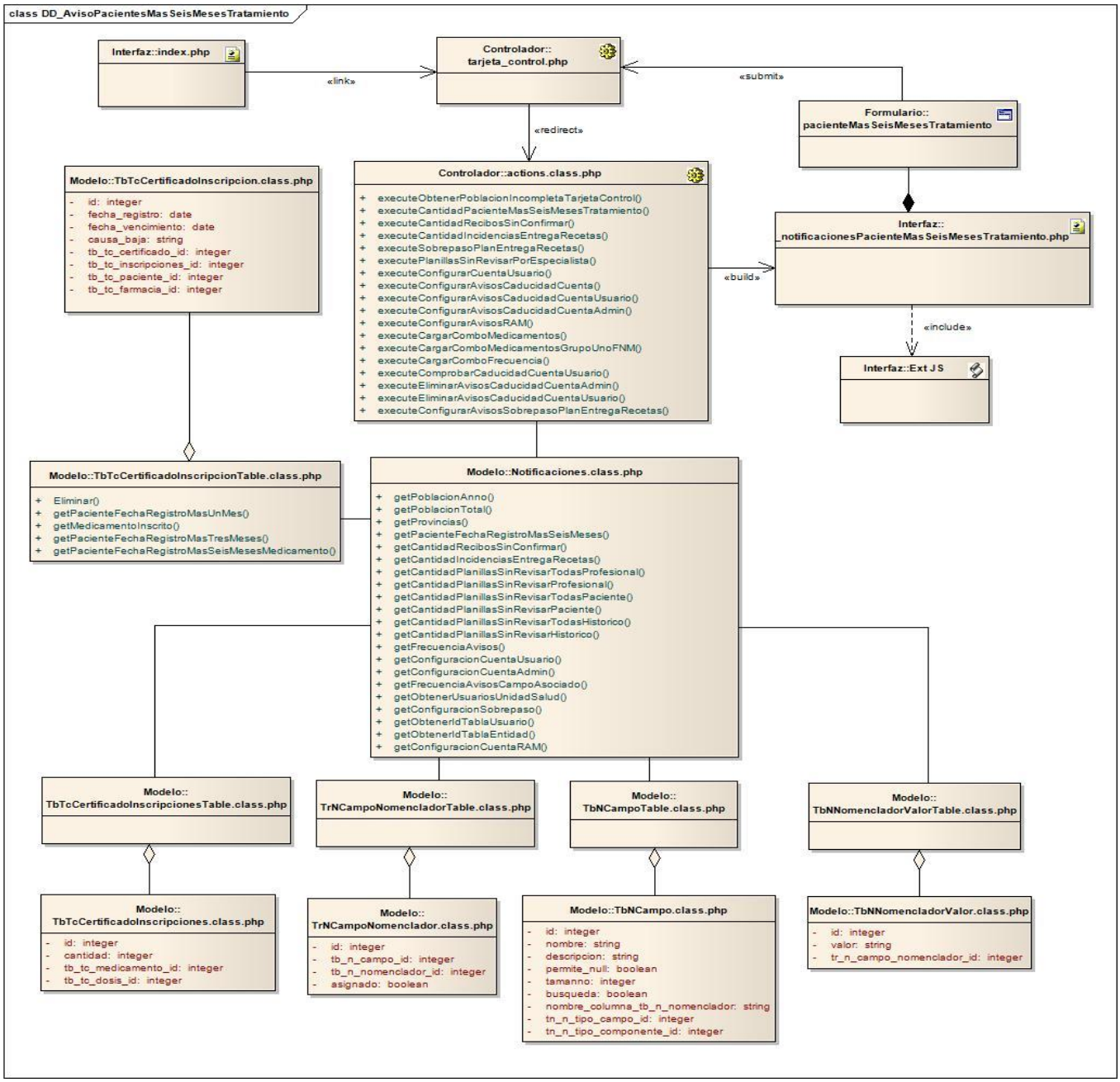


Figura 6. Diagrama de clase del diseño. Caso de uso: Aviso de pacientes con más de seis meses de tratamiento

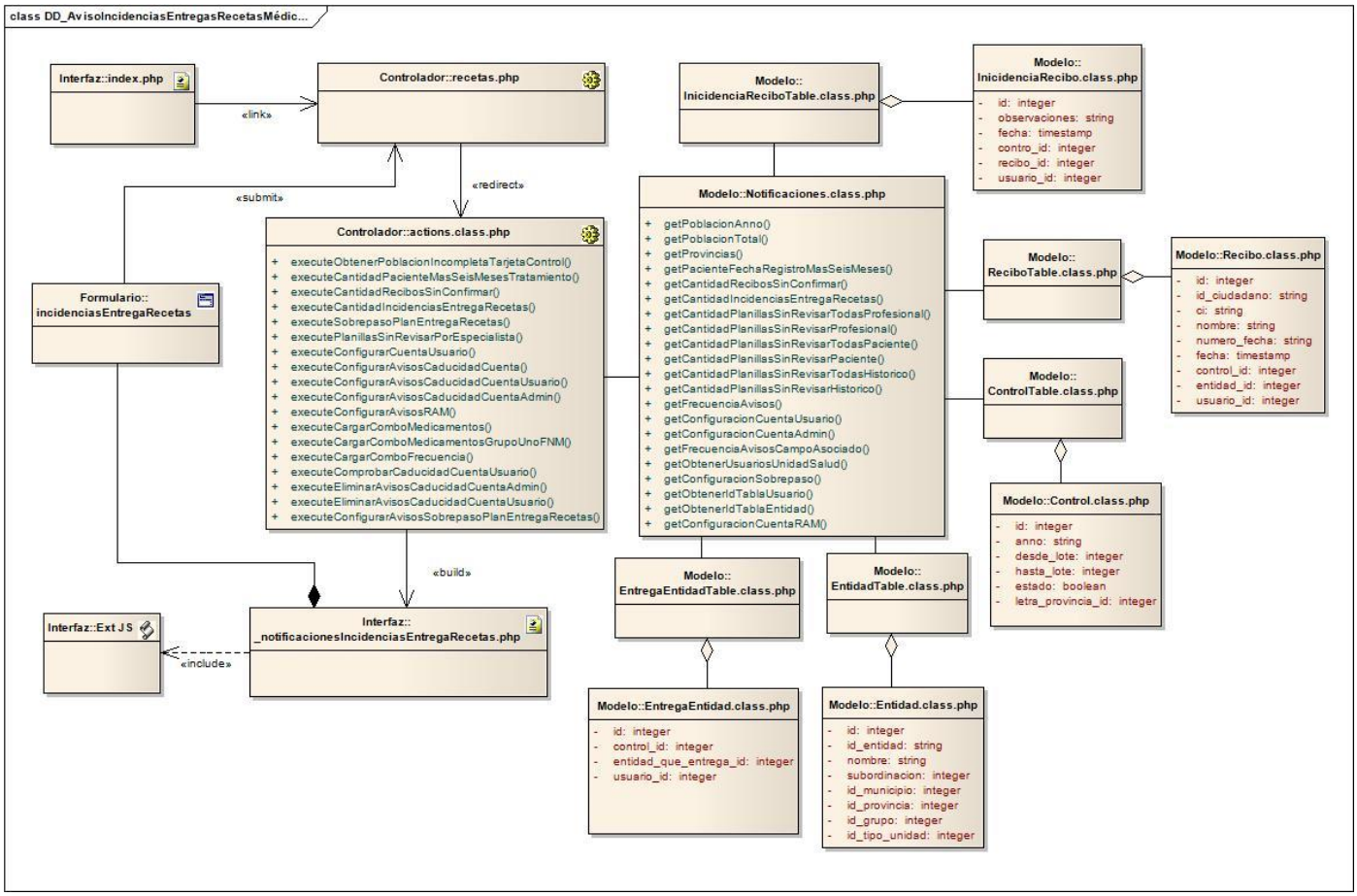


Figura 7. Diagrama de clase del diseño. Caso de uso: Aviso de incidencias en la entrega de recetas

3.1.3 Descripción de las clases del diseño

3.1.3.1 Vistas

Tabla 7. Descripción de vista. Caso de uso: Aviso de registro de población incompleta

Nombre: _notificacionesPoblacionIncompleta.php	
Tipo de clase: interfaz	
Descripción:	Muestra en una interfaz, una lista de avisos en forma de mensajes que informan la existencia de un registro de población incompleta por años.

Tabla 8. Descripción de vista. Caso de uso: Aviso pacientes con más de seis meses de tratamiento

Nombre: _notificacionesPacienteMasSeisMesesTratamiento.php	
Tipo de clase: interfaz	
Descripción:	Muestra en la interfaz de inicio del módulo, un aviso con una lista de mensajes mostrando la cantidad de pacientes con más de seis meses de tratamiento.

Tabla 9. Descripción de vista. Caso de uso: Aviso de incidencias en la entrega de recetas

Nombre: _notificacionesIncidenciasEntregaRecetas.php	
Tipo de clase: interfaz	
Descripción:	Muestra en la interfaz de inicio del módulo, un aviso con una lista de mensajes mostrando la cantidad de incidencias en la entrega de recetas médicas.

3.1.3.2 Controladoras

Tabla 10. Descripción de controladora. Caso de uso: Aviso de registro de población incompleta

Nombre: syntaNotificacionesActions	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	executeObtenerPoblacionIncompletaTarjetaControl()
Descripción:	Permite obtener los registros de población incompleta que existan por año, es decir, comprueba en los registros de población, si existe al menos uno, de los cuatro grupos etarios que hay en cada provincia, con cantidad cero en la población. Cuando esto ocurre

	o cuando no existe un registro de población por cada provincia, el método obtiene el año asociado a este registro.

Tabla 11. Descripción de controladora. Caso de uso: Aviso de pacientes con más de seis meses de tratamiento

Nombre: syntaNotificacionesActions	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	executeCantidadPacienteMasSeisMesesTratamiento()
Descripción:	Permite obtener la cantidad de pacientes que respecto a la fecha actual, lleven más de seis meses con tratamiento de algún medicamento.

Tabla 12. Descripción de controladora. Caso de uso: Aviso de incidencias en la entrega de recetas

Nombre: syntaNotificacionesActions	
Tipo de clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	executeCantidadIncidenciasEntregaRecetas()
Descripción:	Permite obtener la cantidad de incidencias en la entrega de recetas médicas registradas en el sistema.

3.1.3.3 Entidades

Tabla 13. Descripción de entidad. Caso de uso: Aviso de registro de población incompleta

Nombre: TbTcPoblaciones	
Tipo de clase: entidad	
Atributo	**Tipo**
id	integer
grupo_uno	integer
grupo_dos	integer
grupo_tres	integer
grupo_cuatro	integer
anno	integer
tb_tc_provincia_id	integer

Tabla 14. Descripción de entidad. Caso de uso: Aviso de pacientes con más de seis meses de tratamiento

Nombre: TbTcCertificadoInscripcion	
Tipo de clase: entidad	
Atributo	**Tipo**
id	integer
fecha_registro	date
fecha_vencimiento	date

causa_baja	string
tb_tc_certificado_id	integer
tb_tc_inscripciones_id	integer
tb_tc_paciente_id	integer
tb_tc_farmacia_id	integer

Tabla 15. Descripción de entidad. Caso de uso: Aviso de incidencias en la entrega de recetas

Nombre: IncidenciaRecibos	
Tipo de clase: entidad	
Atributo	**Tipo**
id	integer
observaciones	string
fecha	timestamp
control_id	integer
recibo_id	integer
usuario_id	integer

3.2 Patrón arquitectónico

El patrón Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos, estos son: (Pressman, 2005)

- Modelo: contiene todo el contenido específico de la aplicación y la lógica de procesamiento e incluye todos los objetos del contenido, el acceso a fuentes de datos/información externas y toda la funcionalidad de procesamiento que son específicos de la aplicación.
- Vista: contiene todas las funciones específicas de la interfaz y habilita la presentación del contenido y la lógica de procesamiento e incluye todos los objetos de contenido, acceso a fuentes de datos/información externas y toda la funcionalidad de procesamiento requerida por el usuario final.
- Controlador: gestiona el acceso al modelo y a la vista y coordina el flujo de datos entre ellos.

Symfony implementa este patrón clásico del diseño web, trayendo como ventajas, que posibilita tener diferentes vistas para un mismo modelo y además mejora la reusabilidad por medio del desacople entre la vista y el modelo. La estructura en que Symfony implementa el MVC es que en el modelo se encuentra la capa de abstracción a la base de datos y el acceso a los datos; en la vista se podrá encontrar el código de la presentación de la vista, el layout y las plantillas; quedando el controlador, que cuenta con el controlador frontal, que es único para cada aplicación, y las acciones, que incluyen el código específico del controlador de cada página. En el Anexo 4 del presente documento, se encuentra el diagrama de representación del patrón MVC en Symfony.

3.3 Patrones de diseño

Los patrones de diseño son soluciones bien documentadas que se emplean para dar solución a nuevos problemas apoyados en la experiencia de haberlas utilizado con éxito en el pasado. (Almeira, y otros, 2007) Para el diseño de esta solución se emplean los patrones generales de software para asignación de responsabilidades (GRASP, por sus siglas en inglés), con el fin de evitar distintos problemas que pudiesen aparecer durante el desarrollo de la solución, como puede ser, la existencia de una clase con mucha dependencia de otra para ejecutar métodos, lo que pudiese traer consigo, que una modificación de una clase, repercuta en muchas otras. A continuación se describen los patrones GRASP a utilizar.

3.3.1 Patrones GRASP utilizados

- **Experto:** propone asignar una responsabilidad al experto en información, o sea la clase que posee la información necesaria para cumplir con la responsabilidad. (Larman, 1999) En las clases de la capa Modelo se evidencia este patrón, estas poseen un grupo de funcionalidades que están relacionadas

directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

- **Bajo Acoplamiento:** propone asignar una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. (Larman, 1999) Este patrón ayuda a que si existiese una modificación en una clase, repercuta lo menos posible en otras. Se evidencia este patrón en las clases que implementan la lógica del negocio y de acceso a datos que se encuentran en el modelo. Por ejemplo, la clase Notificaciones.class.php implementa la lógica del negocio, esta no tiene asociación con las clases de la capa Vista, lo que proporciona que la dependencia en este caso sea baja.

- **Alta Cohesión:** asignar una responsabilidad de modo que la cohesión siga siendo alta. En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. (Larman, 1999). Con el uso de Symfony para el desarrollo de la presente investigación, se logra una asignación de responsabilidades con una alta cohesión. Este patrón se evidencia en las acciones, por ejemplo en la clase syntaNotificacionesActions. Esta realiza importantes funciones colaborando con otras clases para instanciar objetos y acceder a métodos, siendo la misma la responsable de definir las acciones para las plantillas. El uso de este patrón permite una mejor claridad y facilidad con el que se entiende el diseño además de simplificar el mantenimiento y las mejoras que se quieran realizar en un futuro.

- **Controlador:** plantea asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. (Larman, 1999) La clase syntaNotificacionesActions evidencia el empleo de este patrón, la misma recibe los eventos que se generen en la capa Vista, accede a las clases de la capa Modelo para consultar información de la base de datos, luego procesa la información y envía un resultado a la vista que le corresponda.

3.3.2 Patrones GoF utilizados

Los patrones GoF describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. Están distribuidos en las siguientes categorías: creacionales, estructurales y de comportamiento. (Guerrero, y otros, 2012)

- **Singleton:** el patrón Singleton pertenece a la categoría de creacionales. Este patrón permite que una clase sólo tenga una instancia y proporciona un punto de acceso global a ésta instancia. Se puede evidenciar su uso en la clase perteneciente a la capa Modelo, Notificaciones.class.php, en la cual se crea un método que contiene el único objeto de esta, permitiendo así, que desde otras clases se accedan a otros métodos de la clase Notificaciones.class.php, utilizando este método.

- **Decorador:** el patrón Decorador pertenece a la categoría de estructurales. Este patrón permite agregar funcionalidades dinámicamente. En la solución se encuentra el archivo nombrado layout.php que es el que contiene el layout (plantilla global). Este archivo, guarda el código HTML que es común en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, es decir, el layout decora la plantilla. Este procedimiento es como se implementa el patrón Decorador.

Capítulo 4: Implementación

Introducción

En el capítulo se realiza el modelo de datos y su descripción correspondiente, así como se implementan las clases definidas en el capítulo anterior en términos de componentes. Se conforma el modelo de implementación, modelando los artefactos, diagrama de componentes y diagrama de despliegue, dando una visión de cómo quedará construida y distribuida la solución. Además se plantearán los estándares de codificación a utilizar durante la implementación.

4.1 Modelo de Datos

El modelo de datos es una estructura abstracta que se centra en el planteamiento del desarrollo de aplicaciones de cómo se almacenarán los datos y cómo se accederá a ellos. Determina la estructura de la información, con el objetivo de mejorar la comunicación y la precisión en aplicaciones que usan e intercambian datos. Estos modelos deben tener una única interpretación. (Definicion de, 2013)

Figura 8. Modelo de Datos

En el modelo de datos de la figura anterior, se representan las tablas que se crearon para la gestión de avisos en el Sistema para el control Farmacológico. La tabla TbConfAvisos obtiene la información de la frecuencia y las entidades del módulo Nomenclador y los usuarios del módulo Administración. La tabla TbConfAvisosRAM obtiene la información de los medicamentos y las reacciones adversas a medicamentos del módulo Nomenclador, y la tabla TbConfSobrepasoPlanEntrega obtiene la información de las entidades y de los usuarios del módulo Control de Recetas Médicas, y este a su vez lo obtiene del Sistema para la Autenticación, Autorización y Auditoría (SAAA) del Registro Informatizado de Salud (RIS).

4.1.1 Descripción de las tablas del modelo de datos

Tabla 16. Descripción de la tabla TbAConfAvisos

Nombre: TbAConfAvisos		
Descripción: Almacena los datos relacionados con los parámetros de configuración comunes para los avisos.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador único de la tabla.
cantidad	integer	Valor que indica el parámetro cantidad que se establece en la configuración para ser utilizado en distintos avisos.
frecuencia_id	integer	Identificador del campo frecuencia que se establece en la configuración.
tb_a_usuario_id	integer	Identificador del usuario que realiza la configuración.
tb_a_entidad_id	integer	Identificador de la entidad al que pertenece el usuario.
tipo_de_aviso	integer	Valor que indica el parámetro tipo de aviso.

Tabla 17. Descripción de la tabla TbConfiguracionAvisosRAM

Nombre: TbConfiguracionAvisosRAM
Descripción: Almacena los datos relacionados con las configuraciones de avisos que se establecen en el módulo RAM.

Atributo	Tipo	Descripción
id	integer	Identificador único de la tabla.
tb_a_conf_avisos_id	integer	Identificador de la tabla TbAConfAvisosParam.
tb_a_medicamento_id	integer	Identificador único para el registro de los medicamentos nomenclados para las RAM.
Estado	integer	Valor que indica si la configuración de un aviso está activo o no.
tb_a_ram_id	string	Almacena el identificador de la reacción adversa provocada por un medicamento.

Tabla 18. Descripción de la tabla TbConfiguracionAvisoCaducidadCuenta

Nombre: TbConfiguracionCaducidadCuenta		
Descripción: Almacena los datos relacionados con las configuraciones que se establecen para los avisos de caducidad de la cuenta del usuario y del administrador.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador único de la tabla.
tb_a_conf_avisos_id	integer	Identificador de la tabla TbAConfAvisosParam.
nota_de_contacto	string	Almacena los datos de contacto del administrador del sistema.
tipo_de_configuracion	integer	Valor que indica si la configuración del aviso es para un usuario o un administrador.

Tabla 19. Descripción de la tabla TbConfiguracionSobrepasoPlanEntrega

Nombre: TbConfiguracionSobrepasoPlanEntrega
Descripción: Almacena los datos relacionados con la configuración que se establece para el aviso de sobrepaso del

plan de entrega de recetas médicas.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador único de la tabla.
frecuencia	string	Valor que indica el parámetro frecuencia que se establece en la configuración.
cantidad	integer	Valor que indica el parámetro cantidad que se establece en la configuración
usuario_id	integer	Valor único que identifica a un usuario en la tabla Usuario.
entidad_id	integer	Valor único que identifica a una entidad en la tabla Entidad.

4.2 Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente y ejecutables. Describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponible en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de los otros. (Jacobson, y otros, 2000)

4.2.1 Diagrama de Despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos. Los nodos representan recursos de cómputos: procesadores o dispositivos de hardware. (Jacobson, y otros, 2000)

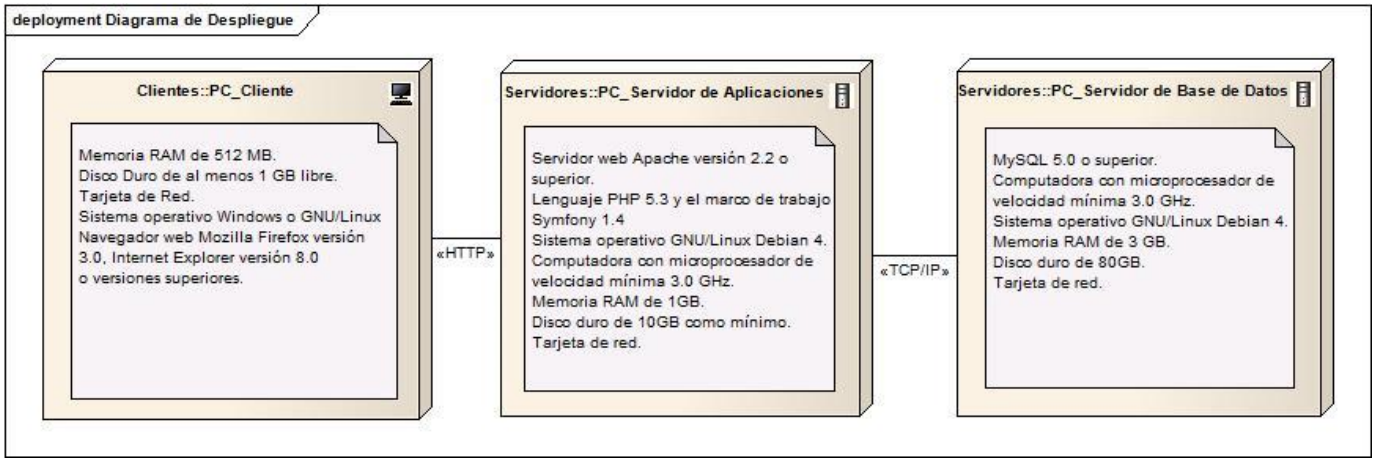


Figura 9. Diagrama de despliegue

Para el despliegue, el usuario debe conectarse al sistema mediante una computadora cliente a través de un navegador web. La comunicación entre el nodo cliente y el servidor de aplicaciones se realiza por el protocolo HTTP y por TCP/IP para establecer la conexión entre el servidor de aplicaciones y el servidor de base de datos.

4.2.2 Diagrama de Componentes

El diagrama de componentes ilustra las piezas del software que conforman un sistema. Tienen un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema. (Sparx System, 2008)

Este diagrama permite mostrar cómo el sistema está divido en componentes y las dependencias entre ellos. Además ayuda a visualizar el camino para la implementación, agrupando en paquetes según un criterio lógico los distintos componentes. A continuación se muestra el diagrama de componentes de la solución a implementar.

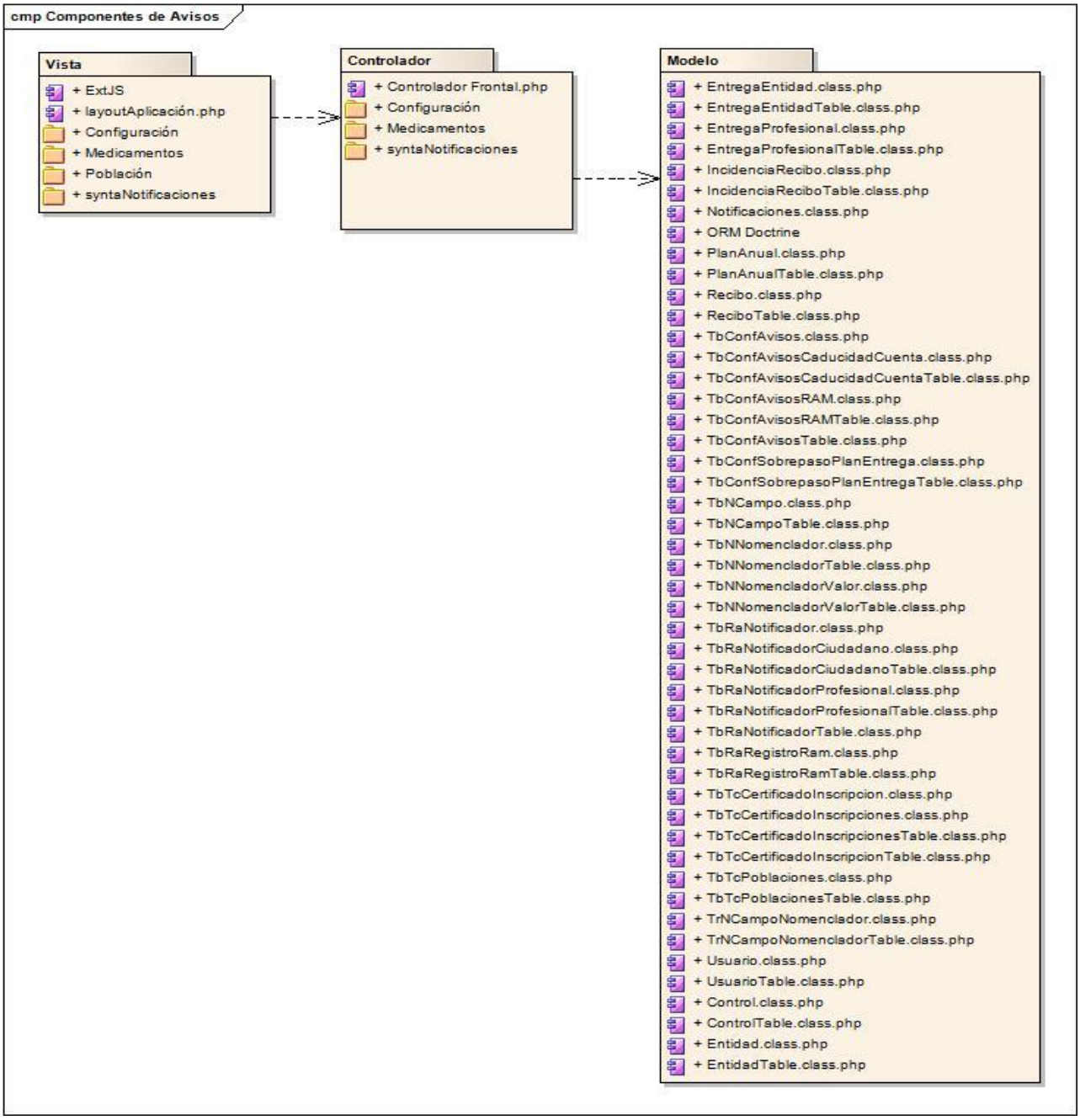


Figura 10. Diagrama de Componentes

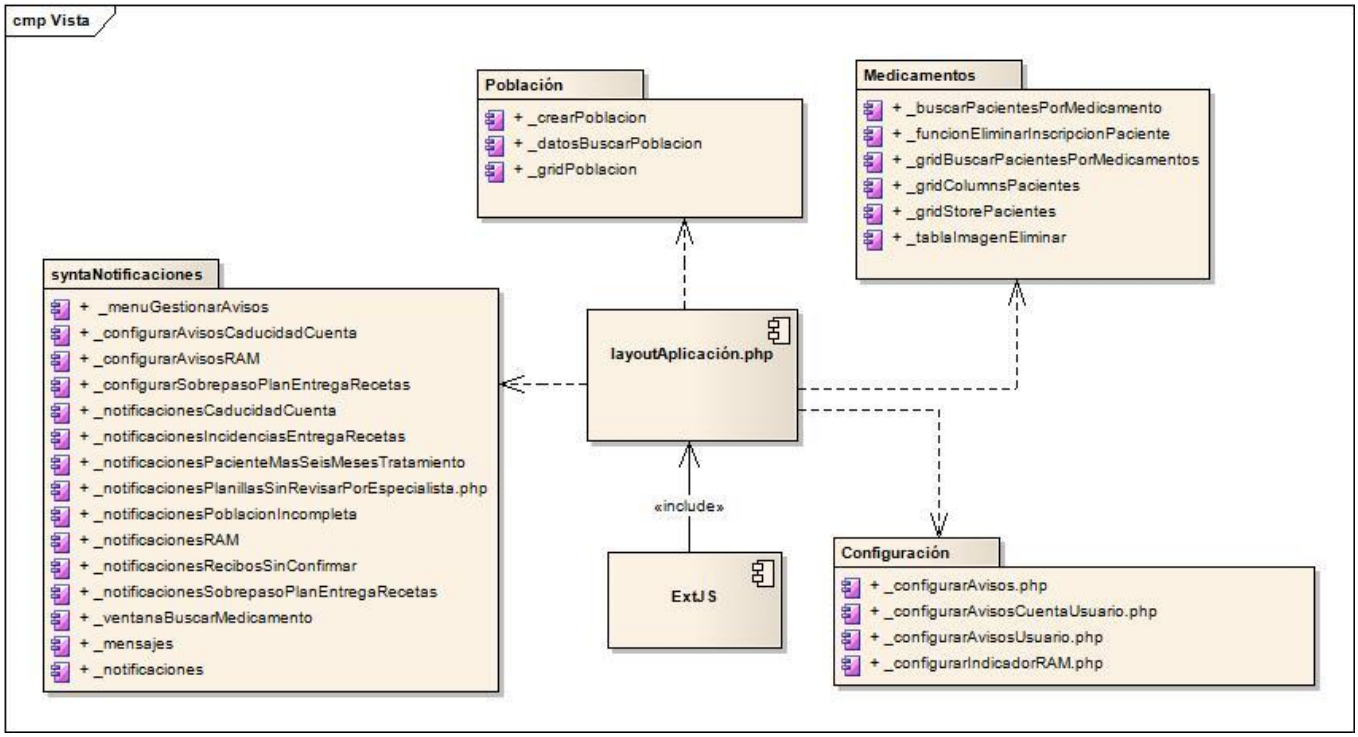


Figura 11. Diagrama de Componentes de la Vista

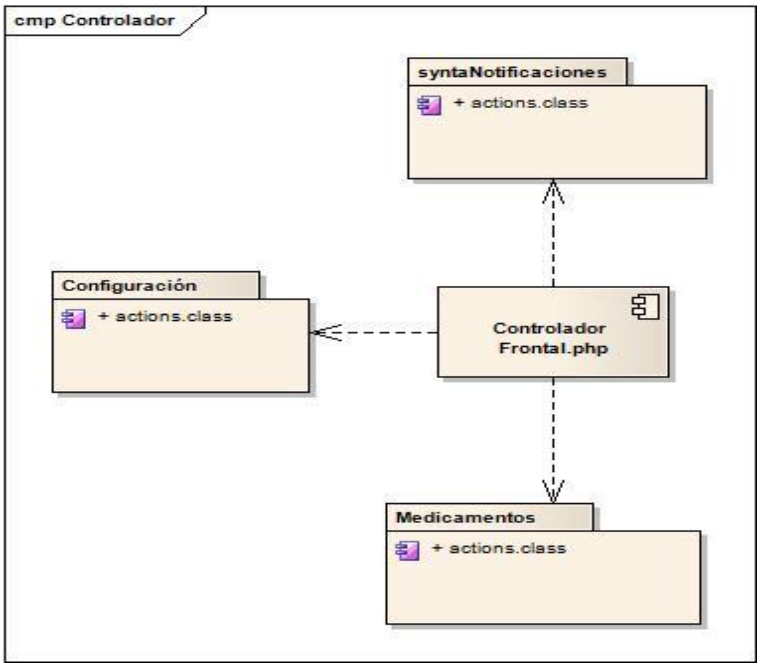


Figura 12. Diagrama de Componentes del Controlador

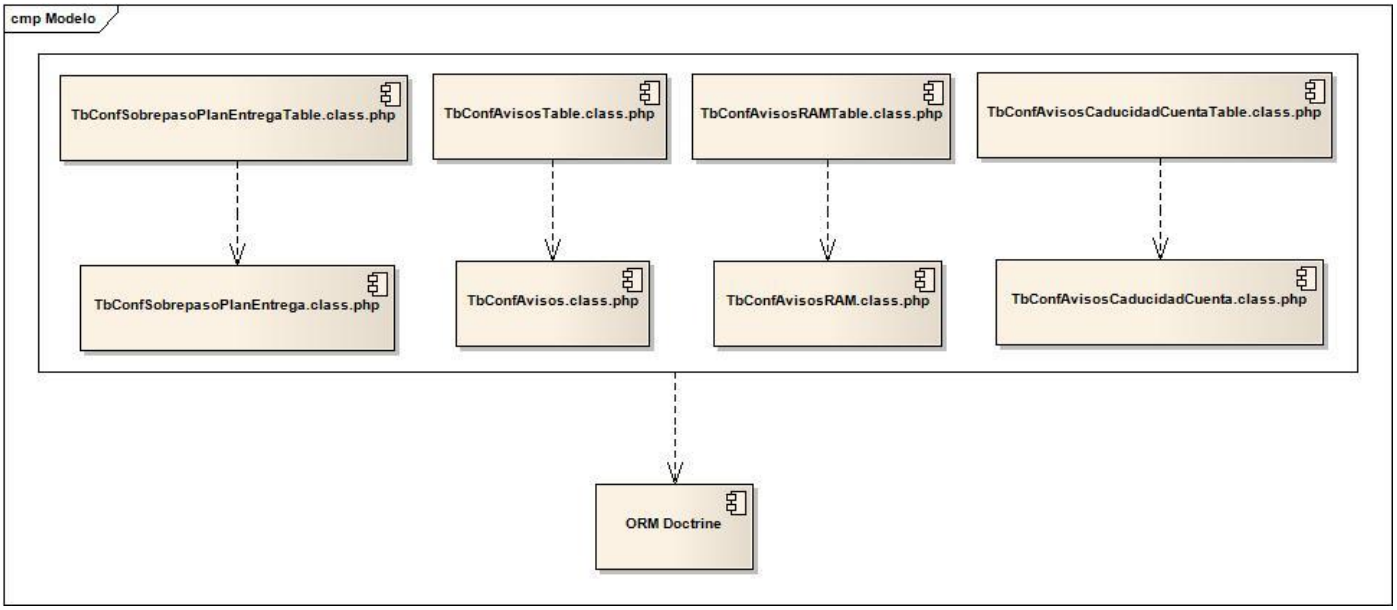


Figura 13. Diagrama de Componentes del Modelo

4.3 Tratamiento de errores

En las funcionalidades se tiene en cuenta el tratamiento de todos los posibles errores que puedan aparecer, por ello, se validan los datos proporcionados por el usuario antes de enviar la información al servidor. Para el tratamiento de errores, se deben realizar en el cliente tantas comprobaciones como sea posible, así se mejoraría el tiempo de respuesta y se reduciría la carga del servidor. Entre los errores más comunes que se pueden presentar durante el desarrollo de una aplicación web, están relacionados fundamentalmente con el trabajo de la base de datos. Para minimizar estos errores, se facilita la introducción de datos en los formularios, siempre que se pueda, con menú de selección, de manera que el usuario solo debe elegir una opción, con esto, los datos que se envían para el servidor estarán validados correctamente y además se logra estandarizar la introducción de datos en los formularios.

Algunas de las funcionalidades también muestran mensajes de información, que son de fácil comprensión para el usuario. Estos se pueden observar cuando se acceden a los formularios de búsquedas y en los mismos, la búsqueda no encuentra información a mostrar. Además se utilizan componentes propios de la librería Ext JS capaces de validar la entrada por el usuario de un tipo específico de datos, ya sea solo valores numéricos o letras.

4.4 Seguridad

La seguridad de un software es de suma importancia para preservar la integridad y confidencialidad de la información. En la solución desarrollada de la presente investigación para gestionar avisos en el Sistema para el control Farmacológico, se tuvo en cuenta estos aspectos de seguridad.

Los usuarios del sistema poseen un rol, una especialidad y pertenecen a una entidad que se subordina a un nivel, esta información de los usuarios es gestionada en el módulo de Administración del sistema, permitiendo garantizar niveles de acceso dentro de la aplicación. Para acceder a los módulos, los usuarios deben primero autenticarse, introduciendo un nombre de usuario y una contraseña, y se comprueba que estos datos sean los correctos, en caso negativo, se mostrará un mensaje de error de acceso.

Los avisos son mostrados a los usuarios teniendo en cuenta, el módulo en que se encuentre trabajando, el nivel de subordinación de la entidad (es decir, Nacional, Provincial, Municipal o Unidad de Salud) y el rol (Visualizador, Editor o Administrador), tratando de minimizar problemas de confidencialidad de la información. Las acciones que conllevan modificación de información, como las funcionalidades de editar o eliminar, solo serán accesible por los usuarios cuyo rol lo permita, en dependencia de la función que realizan en el sistema, de manera que se respete la integridad de los datos.

4.5 Estrategias de codificación. Estándares a utilizar.

Los estándares utilizados en la codificación fueron los siguientes:

- Notación PascalCasing: Está compuesta por tantas palabras como sean necesarias. La primera letra de cada una de las palabras debe ir siempre en mayúsculas.
- Notación CamelCase: Es parecido al PascalCasing con la excepción que la letra inicial del identificador debe estar en minúscula.

4.5.1 Nomenclatura de los métodos

El nombre a emplear para los métodos se escribe con el prefijo “execute” seguido de una palabra en mayúscula, en caso de que sea una palabra compuesta se empleará notación PascalCasing.

Ejemplo: executeCantidadPacienteMasSeisMesesTratamiento.

4.5.2 Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCase. De esta forma es más fácil leer y reconocer el propósito de la misma.

Ejemplo: syntaNotificacionesActions.

4.5.3 Nomenclatura de las variables

El nombre a emplear para las variables se escribe en minúsculas, en caso de que sea un nombre compuesto se empleará notación CamelCase.

Ejemplo: $idUnidadSaludUsuario.

4.5.4 Normas para los comentarios

Se emplean comentarios en todos los métodos, de manera que exista una breve descripción de la funcionalidad que realiza cada método. Sirve de apoyo para una mayor comprensión del código y para facilitar el mantenimiento a lo largo del tiempo.

4.5.5 Estándares

Tabla 20. Estándares a utilizar para la codificación

Estándares	
Líneas en blanco	Se emplean antes y después de métodos, clases y estructuras. Se estila dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función.
Espacios en blanco	Entre operadores lógicos y aritméticos. Se emplean espacios en blanco entre estos operadores para una mayor legibilidad en el código. Ejemplo: $cont = $cont + 1.

Variables	El nombre empleado permite que con sólo leerlo se conozca el propósito de la misma.
Ubicación de los comentarios	Se ubican los comentarios al inicio de cada método y clase. Ejemplo de comentario en una clase: /** syntaNotificaciones actions * * @package synta * @subpackage syntaNotificacionesPlugin * @author Juan Carlos Prado * @version SVN: $Id: actions.class.php 12479 2014-05-19 10:54:40 */
Identación	Se dejan cuatro espacios en blanco desde la instrucción anterior para el inicio de las instrucciones if, for, switch, y foreach.

Conclusiones generales

Con la realización del presente trabajo de diploma se ha cumplido con el objetivo general propuesto, así como con las tareas de la investigación definidas, obteniéndose las siguientes conclusiones:

1. Debido a que los sistemas de gestión de avisos analizados en la investigación no se adecuan a las necesidades del producto Synta, se decidió desarrollar una nueva solución para gestionar avisos en el mismo.
2. Se asimilaron y analizaron las herramientas y tecnologías propuestas por el departamento de Sistemas Especializados en Salud para el desarrollo de la solución, concluyendo que las mismas responden al entorno de desarrollo del sistema Synta posibilitando el objetivo de esta investigación.
3. Se elaboraron los artefactos correspondientes a los flujos de trabajo indicados por la metodología de desarrollo RUP, creando así la documentación necesaria de la investigación que sirvió de guía a los desarrolladores para la implementación de la solución.
4. Se implementó la solución de gestión de avisos en el Sistema para el control Farmacológico, posibilitando que el mismo genere avisos a los usuarios, facilitando así la toma de decisiones.

RECOMENDACIONES

Al concluir el presente trabajo de diploma, el equipo de desarrollo recomienda:

- Analizar el sistema basado en reglas del sistema Synta con el objetivo de identificar nuevos avisos a generar.

Referencias Bibliográficas

Achour, Mehdi, y otros. 2013. *Manual de PHP.* 2013.

Ajax Ya. 2014. Ajax YA. [En línea] 2014. [Citado el: 12 de febrero de 2014.] http://www.ajaxya.com.ar/.

—. 2014. Ajax YA. [En línea] 2014. [Citado el: 12 de febrero de 2014.] www.ajaxya.com.ar.

Almeira, Adriana Sandra y Perez Cavenago, Vanina. 2007. *Arquitectura de Software: Estilos y Patrones.* 2007.

Apache.org. 2014. Apache. [En línea] 2014. [Citado el: 13 de febrero de 2014.] www.apache.org.

Blackboard Learn. 2013. Blackboard Learn. [En línea] 2013. [Citado el: 17 de febrero de 2014.] https://help.blackboard.com/es-es/Learn/9.1_SP_12_and_SP_13/Administrator/190_Tools_Management/Notifications/000_Understanding_the_Framework/000_About_the_Notifications_System#.

Ceria, Santiago. 2013. *Ingeniería de Sofware I.* Buenos Aires : s.n., 2013.

Comusoft. 2010. [En línea] 2010. [Citado el: 19 de febrero de 2014.] http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.

Definición ABC. 2014. Definición ABC. [En línea] 2014. [Citado el: 13 de febrero de 2014.] http://www.definicionabc.com/tecnologia/mysql.php.

—. 2013. Definición ABC. *Definición ABC.* [En línea] 2013. [Citado el: 18 de Diciembre de 2013.] http://www.definicionabc.com/comunicacion/facebook.php.

Definicion de. 2014. Definicion De. [En línea] 2014. [Citado el: 21 de mayo de 2014.] http://definicion.de/notificacion/.

Definición de. 2014. Definición de. [En línea] 2014. [Citado el: 15 de febrero de 2014.] http://definicion.de/evento/.

Definicion de. 2013. Definicion.De. *Definicion.De.* [En línea] 2013. [Citado el: 28 de abril de 2014.] http://definicion.de/modelo-de-datos/.

Desarrollo Web. 2014. Desarrollo Web. [En línea] 2014. [Citado el: 13 de febrero de 2014.] http://www.desarrolloweb.com/wiki/ext-js.html.

Doctrine Project. 2014. [En línea] 2014. [Citado el: 14 de febrero de 2014.] http://www.doctrine-project.org/.

2000 *El proceso unificado de desarrollo de software* MadridAddison Wesley 2000 ISBN: 84-7829-036-2

Estanque Díaz, Diuber, Hidalgo López, Leydis y González Martínez, Yoandy. 2013. *Diseño del mecanismo para alertas y notificaciones del Sistema de Información Hospitalaria ALAS HIS.* 2013. ISBN: 978-959-7213-02-4.

Frederick, Shea, Ramsay, Colin y Cutter Blades, Steve. 2008. *Learning Ext JS.* 2008. ISBN: 978-1-847195-14-2.

García Peñalvo, Dr. Francisco José, Conde González, Miguel Ángel y Bravo Martín, Sergio. 2008. Universidad de Salamanca. [En línea] 16 de Octubre de 2008. [Citado el: 26 de marzo de 2014.] http://ocw.usal.es/ensenanzas-tecnicas/ingenieria-del-software/contenidos/Tema6-DOO-1pp.pdf.

González, Dr. Victoria Ramos. 2007. *www.coit.es.* [En línea] julio de 2007. [Citado el: 21 de noviembre de 2013.] www.coit.es/publicaciones/bit/bit163/41-45.pdf.

Guerrero, Carlos, Gutiérrez, Luz y Suárez, Johanna. 2012. Scielo. *Scielo.* [En línea] noviembre de 2012. [Citado el: 24 de abril de 2014.] http://www.scielo.cl/scielo.php?pid=S0718-07642013000300012&script=sci_arttext.

Hensgen, Paul. 2005. Manual de Umbrello UML Modeller. [En línea] 2005. [Citado el: 24 de 3 de 2014.] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar. 2006. *Metodología de la Investigación.* s.l. : McGraw Hill, 2006. ISBN: 970-10-5753-8.

Larman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Naucalpan de Juárez, México : Prentice Hall, 1999. ISBN: 970-17-0261-1.

Lorenzo, Guillermo Suarez y Hernandez Cárdenaz, Yurlenis. 2011. Repositorio Institucional. *Repositorio Institucional.* [En línea] 2011. [Citado el: 10 de diciembre de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_03976_11/1/TD_03976_11.pdf.

Menéndez, Milena Sánchez. 2012. Repositorio Institucional. *Repositorio Institucional.* [En línea] 2012. [Citado el: 10 de diciembre de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/JCE-2012-F324-P187-Ponencia-2823/1/JCE-2012-F324-P187-Ponencia-2823.doc.

MINSAP. 2013. Dirección de Informática y Comunicaciones del MINSAP. [En línea] MINSAP, 2013. [Citado el: 16 de diciembre de 2013.] http://www.di.sld.cu/cms/?page_id=101.

Morales, Ing. Annia Arencibia. 2012. Repositorio Institucional. *Repositorio Institucional.* [En línea] 9 de marzo de 2012. [Citado el: 10 de diciembre de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/4001/1/uciencia-2012-t40-p1620-ponencia-2255.pdf.

MySQL. 2014. MySQL. [En línea] 2014. [Citado el: 13 de febrero de 2014.] https://dev.mysql.com/doc/refman/5.0/es/features.html.

—. 2014. MySQL. [En línea] 2014. [Citado el: 13 de febrero de 2014.] https://dev.mysql.com/doc/refman/5.0/es/introduction.html.

Netbeans. 2014. Netbeans.org. [En línea] 2014. [Citado el: 14 de febrero de 2014.] https://netbeans.org/features/index.html.

Olivera, Ángel Gabriel y Novelo, Carolina. 2010. [En línea] 6 de septiembre de 2010. [Citado el: 17 de marzo de 2014.] http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales.

Open Suse.org. 2014. Open Suse. [En línea] 2014. [Citado el: 13 de febrero de 2014.] http://es.opensuse.org/Apache.

Ortí, Consuelo Belloch. 2012. Universidad de Valencia. [En línea] 2012. [Citado el: 25 de febrero de 2014.] http://www.uv.es/~bellochc/pdf/pwtic1.pdf.

Pérez Santiesteban, Diana Rosa, y otros. 2013. Módulo de notificaciones y alertas del sistema de gestión universitaria. [En línea] 2013. [Citado el: 18 de febrero de 2014.] http://www.informatica2013.sld.cu/index.php/informaticasalud/2013/paper/view/160/174.

Pérez, Javier Eguíluz. 2009. *Introducción a Javascrip.* 2009.

Pérez, Javier Eguiluz. 2013. Libros Web. [En línea] 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es/ajax/capitulo_1.html.

phpMyAdmin. 2014. phpMyAdmin. [En línea] 2014. [Citado el: 18 de febrero de 2014.] http://www.phpmyadmin.net/home_page/index.php.

Potencier, Fabien y Zaninotto, François. 2010. Libros Web. [En línea] 2010. [Citado el: 12 de febrero de 2014.] http://librosweb.es/symfony_1_4/capitulo_1/symfony_en_pocas_palabras.html.

Pressman, Roger S. 2005. *Ingenieria de Software Un enfoque práctico.* 6ta Edición. s.l. : McGraw Hill, 2005. ISBN: 9701054733.

2013. Programación. [En línea] 2013. [Citado el: 24 de febrero de 2014.] http://www.programacion.com/articulo/por_que_elegir_php_143.

RAE. 2001. Real Academia Española. *sitio web de la Real Academia Española.* [En línea] Real Academia Española, 2001. [Citado el: 29 de noviembre de 2013.] http://lema.rae.es/drae/srv/search?id=QHTyWym2eDXX2LBnET9H.

—. 2001. Sitio web de la Real Academia Española. [En línea] 2001. [Citado el: 21 de mayo de 2014.] http://lema.rae.es/drae/?val=Aviso.

Reynoso, Carlos y Kiccillof, Nicolás. 2004. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] 2004. [Citado el: 24 de marzo de 2014.] http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp.

Rodríguez, Alfredo Sánchez y Pompa Sourd, Frank. 2007. Direccion de Informática y Comunicaciones del MINSAP. [En línea] 2007. [Citado el: 30 de abril de 2014.] http://www.di.sld.cu/cms/wp-content/uploads/2013/12/Normas-para-el-desarrollo-de-aplicaciones-informaticas-para-la-Salud-en-Cuba.pdf.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2007. *El lenguaje unificado de modelado. Manual de Referencia.* 2007. 9788478290871.

Softqanetwork. 2009. Softqanetwork.com. [En línea] 2009. [Citado el: 13 de marzo de 2013.] http://www.softqanetwork.com/requisitos-no-funcionales-nfr.

Sparx System. 2008. Sparx System. [En línea] 2008. [Citado el: 24 de abril de 2014.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html.

Sparx Systems. 2008. Sparx Systems.es. [En línea] Sparx Systems, 2008. [Citado el: 26 de febrero de 2014.] http://www.sparxsystems.es/New/products/ea.html.

Symfony.es. 2014. Symfony.es. [En línea] 2014. [Citado el: 12 de febrero de 2014.] http://symfony.es/que-es-symfony.

—. 2009. Symfony.es. [En línea] 2009. [Citado el: 14 de febrero de 2014.] http://symfony.es/noticias/2009/10/05/netbeans-ya-incluye-soporte-para-symfony/.

UCI. 2013. Gestpro. [En línea] 2013. [Citado el: 31 de marzo de 2014.] http://gespro.cesim.prod.uci.cu/.

VOX. 2011. Diccionarios.com. [En línea] Larousse Editorial, 2011. [Citado el: 15 de febrero de 2014.] http://www.diccionarios.com/detalle.php?palabra=evento&Buscar.x=0&Buscar.y=0&dicc_100=on&palabra2=.

Werdmuller, Ben. 2010. *Build a web-based notification tool with XMPP.* 2010.

Bibliografía

Achour, Mehdi, y otros. 2013. *Manual de PHP.* 2013.

Ajax Ya. 2014. Ajax YA. [En línea] 2014. [Citado el: 12 de febrero de 2014.] http://www.ajaxya.com.ar/.

—. 2014. Ajax YA. [En línea] 2014. [Citado el: 12 de febrero de 2014.] www.ajaxya.com.ar.

Almeira, Adriana Sandra y Perez Cavenago, Vanina. 2007. *Arquitectura de Software: Estilos y Patrones.* 2007.

Apache.org. 2014. Apache. [En línea] 2014. [Citado el: 13 de febrero de 2014.] www.apache.org.

Blackboard Learn. 2013. Blackboard Learn. [En línea] 2013. [Citado el: 17 de febrero de 2014.] https://help.blackboard.com/es-es/Learn/9.1_SP_12_and_SP_13/Administrator/190_Tools_Management/Notifications/000_Understanding_the_Framework/000_About_the_Notifications_System#.

—. 2013. Blackboard Learn. [En línea] Blackboard Learn.inc, 2013. [Citado el: 17 de febrero de 2014.] http://www.blackboard.com.

Carbonnelle, Pierre. 2014. PYPL PopularitY of Programming Language index. [En línea] 2014. [Citado el: 24 de febrero de 2014.] https://sites.google.com/site/pydatalog/pypl/PyPL-PopularitY-of-Programming-Language.

Ceria, Santiago. 2013. *Ingeniería de Sofware I.* Buenos Aires : s.n., 2013.

Comusoft. 2010. [En línea] 2010. [Citado el: 19 de febrero de 2014.] http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.

Definición ABC. 2014. Definición ABC. [En línea] 2014. [Citado el: 13 de febrero de 2014.] http://www.definicionabc.com/tecnologia/mysql.php.

—. 2013. Definición ABC. *Definición ABC.* [En línea] 2013. [Citado el: 18 de Diciembre de 2013.] http://www.definicionabc.com/comunicacion/facebook.php.

Definicion de. 2014. Definicion De. [En línea] 2014. [Citado el: 21 de mayo de 2014.] http://definicion.de/notificacion/.

Definición de. 2014. Definición de. [En línea] 2014. [Citado el: 15 de febrero de 2014.] http://definicion.de/evento/.

—. Definición de modelo de datos - Qué es, Significado y Concepto. [En línea] [Citado el: 21 de Abril de 2014.] http://definicion.de/modelo-de-datos/.

Definicion de. 2013. Definicion.De. *Definicion.De.* [En línea] 2013. [Citado el: 28 de abril de 2014.] http://definicion.de/modelo-de-datos/.

Desarrollo Web. 2014. Desarrollo Web. [En línea] 2014. [Citado el: 13 de febrero de 2014.] http://www.desarrolloweb.com/wiki/ext-js.html.

Doctrine Project. 2014. [En línea] 2014. [Citado el: 14 de febrero de 2014.] http://www.doctrine-project.org/.

Eguiluz Pérez, Javier. 2013. Libros Web. *Introducción a AJAX.* [En línea] 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es/ajax/capitulo_1.html.

2000*El proceso unificado de desarrollo de software*MadridAddison Wesley2000ISBN: 84-7829-036-2

Estanque Díaz, Diuber, Hidalgo López, Leydis y González Martínez, Yoandy. 2013. *Diseño del mecanismo para alertas y notificaciones del Sistema de Información Hospitalaria ALAS HIS.* 2013. ISBN: 978-959-7213-02-4.

Frederick, Shea, Ramsay, Colin y Cutter Blades, Steve. 2008. *Learning Ext JS.* 2008. ISBN: 978-1-847195-14-2.

García Peñalvo, Dr. Francisco José, Conde González, Miguel Ángel y Bravo Martín, Sergio. 2008. Universidad de Salamanca. [En línea] 16 de Octubre de 2008. [Citado el: 26 de marzo de 2014.] http://ocw.usal.es/ensenanzas-tecnicas/ingenieria-del-software/contenidos/Tema6-DOO-1pp.pdf.

González, Dr. Victoria Ramos. 2007. *www.coit.es.* [En línea] julio de 2007. [Citado el: 21 de noviembre de 2013.] www.coit.es/publicaciones/bit/bit163/41-45.pdf.

Guerrero, Carlos, Gutiérrez, Luz y Suárez, Johanna. 2012. Scielo. *Scielo.* [En línea] noviembre de 2012. [Citado el: 24 de abril de 2014.] http://www.scielo.cl/scielo.php?pid=S0718-07642013000300012&script=sci_arttext.

Hensgen, Paul. 2005. Manual de Umbrello UML Modeller. [En línea] 2005. [Citado el: 24 de 3 de 2014.] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2011. *El Proceso de Investigación Científica.* Ciudad de La Habana : Editorial Universitaria, 2011. ISBN: 978-959-16-1307-3.

Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar. 2006. *Metodología de la Investigación.* s.l. : McGraw Hill, 2006. ISBN: 970-10-5753-8.

JavaScript Ya. JavaScript Ya. [En línea] [Citado el: 18 de febrero de 2014.] http://www.javascriptya.com.ar/temarios/descripcion.php?cod=2.

Larman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Naucalpan de Juárez, México : Prentice Hall, 1999. ISBN: 970-17-0261-1.

Lorenzo, Guillermo Suarez y Hernandez Cárdenaz, Yurlenis. 2011. Repositorio Institucional. *Repositorio Institucional.* [En línea] 2011. [Citado el: 10 de diciembre de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_03976_11/1/TD_03976_11.pdf.

Menéndez, Milena Sánchez. 2012. Repositorio Institucional. *Repositorio Institucional.* [En línea] 2012. [Citado el: 10 de diciembre de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/JCE-2012-F324-P187-Ponencia-2823/1/JCE-2012-F324-P187-Ponencia-2823.doc.

MINSAP. 2013. Dirección de Informática y Comunicaciones del MINSAP. [En línea] MINSAP, 2013. [Citado el: 16 de diciembre de 2013.] http://www.di.sld.cu/cms/?page_id=101.

Morales, Ing. Annia Arencibia. 2012. Repositorio Institucional. *Repositorio Institucional.* [En línea] 9 de marzo de 2012. [Citado el: 10 de diciembre de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/4001/1/uciencia-2012-t40-p1620-ponencia-2255.pdf.

MySQL. 2014. MySQL. [En línea] 2014. [Citado el: 13 de febrero de 2014.] https://dev.mysql.com/doc/refman/5.0/es/features.html.

—. 2014. MySQL. [En línea] 2014. [Citado el: 13 de febrero de 2014.] https://dev.mysql.com/doc/refman/5.0/es/introduction.html.

Netbeans. 2014. Netbeans.org. [En línea] 2014. [Citado el: 14 de febrero de 2014.] https://netbeans.org/features/index.html.

Olivera, Ángel Gabriel y Novelo, Carolina. 2010. [En línea] 6 de septiembre de 2010. [Citado el: 17 de marzo de 2014.] http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales.

Open Suse.org. 2014. Open Suse. [En línea] 2014. [Citado el: 13 de febrero de 2014.] http://es.opensuse.org/Apache.

Ortí, Consuelo Belloch. 2012. Universidad de Valencia. [En línea] 2012. [Citado el: 25 de febrero de 2014.] http://www.uv.es/~bellochc/pdf/pwtic1.pdf.

Pérez Santiesteban, Diana Rosa, y otros. 2013. Módulo de notificaciones y alertas del sistema de gestión universitaria. [En línea] 2013. [Citado el: 18 de febrero de 2014.] http://www.informatica2013.sld.cu/index.php/informaticasalud/2013/paper/view/160/174.

Pérez, Javier Eguíluz. 2009. *Introducción a Javascrip.* 2009.

Pérez, Javier Eguiluz. 2013. Libros Web. [En línea] 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es/ajax/capitulo_1.html.

Pérez, Javier Eguíluz. 2013. Libros Web. [En línea] 14 de marzo de 2013. [Citado el: 11 de febrero de 2014.] http://librosweb.es.

phpMyAdmin. 2014. phpMyAdmin. [En línea] 2014. [Citado el: 18 de febrero de 2014.] http://www.phpmyadmin.net/home_page/index.php.

Potencier, Fabien y Zaninotto, François. 2010. Libros Web. [En línea] 2010. [Citado el: 12 de febrero de 2014.] http://librosweb.es/symfony_1_4/capitulo_1/symfony_en_pocas_palabras.html.

Pressman, Roger S. 2005. *Ingenieria de Software Un enfoque práctico.* 6ta Edición. s.l. : McGraw Hill, 2005. ISBN: 9701054733.

2013. Programación. [En línea] 2013. [Citado el: 24 de febrero de 2014.] http://www.programacion.com/articulo/por_que_elegir_php_143.

RAE. 2001. Real Academia Española. *sitio web de la Real Academia Española.* [En línea] Real Academia Española, 2001. [Citado el: 29 de noviembre de 2013.] http://lema.rae.es/drae/srv/search?id=QHTyWym2eDXX2LBnET9H.

—. 2001. Sitio web de la Real Academia Española. [En línea] 2001. [Citado el: 21 de mayo de 2014.] http://lema.rae.es/drae/?val=Aviso.

Reynoso, Carlos y Kiccillof, Nicolás. 2004. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] 2004. [Citado el: 24 de marzo de 2014.] http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp.

Rodríguez, Alfredo Sánchez y Pompa Sourd, Frank. 2007. Direccion de Informática y Comunicaciones del MINSAP. [En línea] 2007. [Citado el: 30 de abril de 2014.] http://www.di.sld.cu/cms/wp-content/uploads/2013/12/Normas-para-el-desarrollo-de-aplicaciones-informaticas-para-la-Salud-en-Cuba.pdf.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2007. *El lenguaje unificado de modelado. Manual de Referencia.* 2007. 9788478290871.

Sencha.com. 2014. Sencha.com. [En línea] 2014. [Citado el: 15 de mayo de 2014.] http://www.sencha.com/products/extjs/license/.

Softqanetwork. 2009. Softqanetwork.com. [En línea] 2009. [Citado el: 13 de marzo de 2013.] http://www.softqanetwork.com/requisitos-no-funcionales-nfr.

Sommerville, Ian. 2005. *Ingeniería del Software.* Madrid : Adisson Wesley, 2005. ISBN: 84-7929-074-5.

Sparx System. 2008. Sparx System. [En línea] 2008. [Citado el: 24 de abril de 2014.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html.

Sparx Systems. 2008. Sparx Systems.es. [En línea] Sparx Systems, 2008. [Citado el: 26 de febrero de 2014.] http://www.sparxsystems.es/New/products/ea.html.

Symfony. 2010. Symfony.es. [En línea] 2010. [Citado el: 24 de febrero de 2014.] http://symfony.es/noticias/2010/02/17/sflive2010-asi-es-symfony-2/.

Symfony.es. 2014. Symfony.es. [En línea] 2014. [Citado el: 12 de febrero de 2014.] http://symfony.es/que-es-symfony.

—. 2009. Symfony.es. [En línea] 2009. [Citado el: 14 de febrero de 2014.] http://symfony.es/noticias/2009/10/05/netbeans-ya-incluye-soporte-para-symfony/.

UCI. 2013. Gestpro. [En línea] 2013. [Citado el: 31 de marzo de 2014.] http://gespro.cesim.prod.uci.cu/.

VOX. 2011. Diccionarios.com. [En línea] Larousse Editorial, 2011. [Citado el: 15 de febrero de 2014.] http://www.diccionarios.com/detalle.php?palabra=evento&Buscar.x=0&Buscar.y=0&dicc_100=on&palabra2=.

Werdmuller, Ben. 2010. *Build a web-based notification tool with XMPP.* 2010.

Glosario de Términos

API: Application Programming Interface (Interfaz de Programación de Aplicaciones), es el conjunto de funciones y procedimientos que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

ASP: Active Server Pages, es una tecnología de Microsoft del tipo "lado del servidor" para páginas web generadas dinámicamente.

BPMN: Notación para el Modelado de Procesos del Negocio.

CASE: Computer Arded Software Engineering (Ingeniería de Software Asistida por Computadoras)

CBM: Cuadro Básico de Medicamentos.

CESIM: Centro de Informática Médica.

CSS: Cascading Style Sheets (Hojas de estilo en cascada), es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML.

CU: Caso de Uso.

DBAL: Capa de abstracción de bases de datos.

DHTML: Dynamic HTML, es el conjunto de técnicas que permiten crear sitios web dinámicos utilizando una combinación de lenguajes como HTML, JavaScript, CSS y un Modelo en Objetos para la Representación de Documentos.

DQL: Doctrine Query Language, lenguaje de consulta que utiliza el sistema de mapeo objeto relacional Doctrine.

GPL: General Public License (Licencia Pública General).

Grails: Es un marco de trabajo libre para desarrollar aplicaciones web.

HQL: Hibernate Query Language, lenguaje de consulta que utiliza el sistema de mapeo objeto relacional Hibernate.

HTML: HyperText Markup Language (Lenguaje de Marcas de Hipertexto).

GLOSARIO DE TÉRMINOS

HTTP: Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto), es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

IDE: Entorno de Desarrollo Integrado.

IIS: Internet Information Services, es un servidor web y un conjunto de servicios para el sistema operativo Windows.

JSP: Java Server Pages, es una tecnología Java que permite generar contenido dinámico para la web.

MINSAP: Ministerio de Salud Pública.

MVC: Modelo-Vista-Controlador

ORM: Object-Relational Mapping (Mapeo de Objetos-Relacional).

PHP: Hypertext Pre-Processor.

Plugin: Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

Ruby: Es un lenguaje de programación interpretado y orientado a objetos.

RUP: Rational Unified Process (El Proceso Unificado de Desarrollo).

SGU: Sistema de Gestión Universitaria.

SNS: Sistema Nacional de Salud.

SMS: Short Message Service, (Servicio de Mensajes Cortos), es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos entre celulares.

SVN: Se refiere a Subversion, un sistema de control de versiones.

TIC: Tecnologías de la Información y las Comunicaciones.

UCI: Universidad de Ciencias Informáticas.

UML: Unified Modeling Language (Lenguaje de Modelado Unificado).

XML: eXtensible Markup Language (Lenguaje de Marcado Extensible).

Anexos

Anexo 1 Entrevista

Entrevista realizada a la MSc. Annia Arencibia Morales, ocupa el cargo de Analista del proyecto Synta.

¿Qué módulos del Sistema para el control Farmacológico necesitan gestión de avisos?

Los módulos de Control de Recetas Médicas, Tarjeta Control, Reacciones Adversas a Medicamentos y Administración son los que necesitan avisos, siendo en los mismos donde existe información que necesita ser conocida por los usuarios de una manera automática.

¿Específicamente qué información necesita ser mostrada mediante avisos en cada módulo?

En los módulos Control de Recetas Médicas, Tarjeta Control, Reacciones Adversas a Medicamentos y Administración existe información, que al alcanzar determinados valores o comportamiento deben ser analizados por el personal especializado para tomar decisiones. Los avisos que se necesitarán por módulos son los siguientes:

Control de Recetas Médicas:

- Aviso ante la ocurrencia de incidencias reportadas en las entregas de recetas médicas.
- Aviso ante la existencia de recibos sin confirmar.
- Aviso con los nombres de los profesionales, entidades, municipios o provincias que sobrepasaron el plan de entregas en dependencia del nivel.

Tarjeta Control:

- Aviso ante la existencia de un registro de población incompleta.
- Aviso de existencia de pacientes con más de seis meses de tratamiento.

Reacciones Adversas a Medicamentos:

- Aviso ante la existencia de planillas del profesional o paciente sin revisar por el especialista.
- Aviso sobre la cantidad de reacciones adversas provocadas por medicamentos.

Administración:

- Aviso al administrador con los nombres de los usuarios próximos a caducar.
- Además en todos los todos los módulos se debe mostrar una aviso al usuario de caducidad de la cuenta.

¿Qué avisos necesitan ser configurables?

En el módulo de Control de Recetas Médicas, necesita ser configurable el aviso de sobrepaso del plan de entrega, de manera que el usuario defina una cantidad límite para el plan de entrega de recetas médicas. En el módulo de Reacciones Adversas a Medicamentos se debe configurar el aviso para mostrar los medicamentos que sobrepasen una cantidad de reacciones adversas, teniendo en cuenta un valor definido por el usuario. En el módulo de Administración se deben configurar los avisos, para mostrarle al usuario el tiempo restante en que caducará la cuenta y para mostrar al administrador los usuarios próximos a caducar.

¿Qué otras funcionalidades se necesitan relacionadas con la gestión de avisos?

En el módulo de Tarjeta Control, además de mostrar el aviso de pacientes con más de seis meses de tratamiento, se necesita una funcionalidad que permita buscar los pacientes inscritos por medicamentos por períodos de: uno, tres y seis meses, de manera que el usuario pueda gestionar esta información de una manera más flexible. Además brindar la posibilidad de eliminar la inscripción de un paciente asociado a un medicamento controlado.

¿De qué manera se requiere que se muestren los avisos?

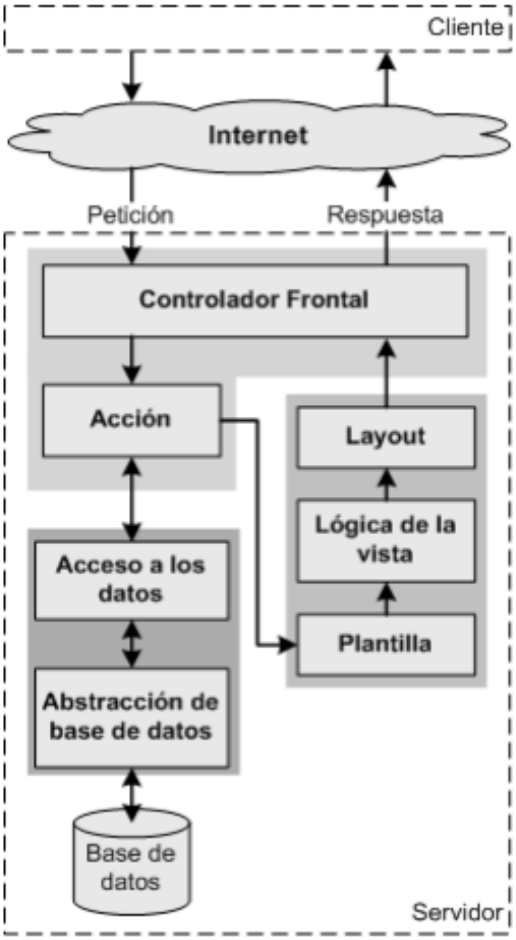
Los avisos se deben mostrar automáticamente cuando el usuario realice el evento de autenticación en el sistema y seleccione el módulo en que va a trabajar. Todos se deben presentar en la pestaña de inicio de cada módulo en forma de mensajes.

Los avisos deben estar asociados a los usuarios en dependencia del módulo, el nivel (nacional, provincial, municipal o unidad de salud), de la entidad a la que pertenecen y al rol (visualizador, editor o administrador) que tengan establecidos en el módulo de Administración.

Anexo 2 Tabla de comparativa de lenguajes de programación.

Posición Feb 2014	Posición Feb 2013	Lenguaje	Búsquedas Feb 2014	Doce meses Tendencias
1	1	Java	26.1 %	-0.8 %
2	2	PHP	13.5 %	-1.2 %
3	5	Python	10.2 %	+0.8 %
4	3	C#	10.0 %	-0.2 %
5	4	C++	8.6 %	-0.2 %
6	6	C	8.2 %	-0.4 %
7	7	Javascript	7.9 %	+0.1 %
8	8	Objective-C	7.0 %	+1.3 %
9	9	Ruby	3.1 %	+0.1 %
10	10	Visual Basic	3.5 %	-0.2 %

Anexo 3 Diagrama del flujo de trabajo de Symfony



Anexo 4 Diagrama de representación del patrón MVC en Symfony