Modelo de Sistema para la Gestión de Pacientes en Hospitales Públicos bajo arquitectura SOA

Gastón Courtois, Pablo Telmo, Andrés Sosto, Maximiliano Bordón, Damián Gargiulo, Matías Lapera, Ezequiel Colonna

{gastoncourtois, pablo.telmo, asosto, maximiliano.bordon, damian.gargiulo, lapera.matias, ezequielcolonna}@gmail.com

Instituto de Sistemas Inteligentes y Enseñanza Experimental de la Robótica. Facultad de Informática, Ciencias de la Comunicación y Técnicas Especiales. Universidad de Morón.

Cabildo 134, (B1708JPD) Morón, Buenos Aires, Argentina
Tel: 54-11-5627-2000 - Fax: 54-11-5627-2002

RESUMEN

En virtud de las principales necesidades que afrontan gran parte de los Hospitales Públicos en Argentina, como la deficiente disponibilidad de información veraz, completa, oportuna y centralizada y los bajos recursos tecnológicos, se ha decidido realizar un modelo de solución informática para el área de cardiología del Hospital Posadas.

El objetivo del mismo es mejorar el conjunto de actividades, métodos, procesos y procedimientos para la prestación de servicios, utilizando los datos ingresados en el sistema con el fin de obtener oportunamente indicadores de gestión, estadísticas vitales en salud, informes de atención y registros individuales de prestación de servicios.

Paralelamente el proyecto tiene el rigor académico correspondiente, en términos de metodologías de investigación, aplicación de heurísticas de diseño y las mejores prácticas de Ingeniería de Software.

Palabras Claves: gestión de pacientes, arquitectura orientada a servicios, ingeniería de usabilidad, escalabilidad, hospitales públicos.

1. INTRODUCCIÓN

En base al relevamiento realizado se ha determinado la necesidad de desarrollar el sistema en un entorno Web, debido a una marcada tendencia de este tipo de aplicaciones a ejecutarse sobre clientes delgados y equipos con baja capacidad de procesamiento.

Otro aspecto a considerar fue el diseño arquitectónico del sistema. Se han investigado distintos tipos de arquitecturas y se determinó la más conveniente a este sistema.

Factores considerados en el diseño y desarrollo son:

- Escalabilidad: en base a este concepto se intentan establecer bases sólidas para el crecimiento del sistema.
- Usabilidad: teniendo en cuenta la diversidad de conocimientos y rotación del personal que interactúa con el sistema, se ha hecho hincapié en lograr una interfaz de usuario amigable y ágil.
- Seguridad: el sistema contempla mecanismos de autenticación y autorización de usuarios, como así también encriptación de datos sensibles.
- Flexibilidad: la configuración de la aplicación permite definir y adaptar ésta a diferentes escenarios, como el soporte para diferentes idiomas, cambios de textos en pantallas y generación dinámica de campos en los módulos necesarios.

2. DISEÑO ARQUITECTÓNICO

La Arquitectura de Software se define como los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura de software debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea de computación.

Además, establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información.

Existen distintos tipos de arquitectura con sus ventajas e inconvenientes, como ser:

- Monolítica: el software se estructura en grupos funcionales muy acoplados.
- Cliente-servidor. el software reparte su lógica en dos partes independientes pero sin reparto claro de funciones.
- Arquitectura de tres capas: se divide en tres capas con un reparto claro de funciones: una capa para la presentación, otra para la lógica de negocios y otra para los datos. Una capa solamente tiene relación con la siguiente.

Luego de investigar los mencionados tipos, se ha optado por utilizar una Arquitectura Orientada a Servicios (SOA)[5], dado que es la que mejor se adapta a este tipo de sistema porque permite interactuar con otras aplicaciones y equipos.

Arquitectura Orientada a Servicios (SOA)

La Arquitectura Orientada a Servicios (Service-oriented architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y dar soporte a las actividades de integración y consolidación.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP[i] y WSDL[ii]) en su implementación. No obstante se puede implementar SOA utilizando cualquier tecnología basada en servicios. Éstos están débilmente acoplados y son altamente interoperables. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación (p.ej., WSDL). La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Java[11] o .NET[10]). Con esta arquitectura, se pretende que los componentes software desarrollados sean reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio .NET podría ser usado por una aplicación Java.

Arquitectura del Sistema

Se ha optado por utilizar una arquitectura SOA dado que el negocio de un Hospital cada vez exige crear aplicaciones más complejas, optimizando los factores tiempo y presupuesto. Crear estas aplicaciones, requiere en muchos casos de funcionalidades ya antes implementadas como parte de otros sistemas.

Exponer procesos de negocio como servicios es la clave de la flexibilidad de la arquitectura. Esto permite que otras piezas de funcionalidad (incluso también implementadas como servicios) hagan uso de otros servicios de manera natural, sin importar su ubicación física. Así el sistema evoluciona con la adición de nuevos servicios y su mejoramiento, donde cada servicio evoluciona de una manera independiente. La Arquitectura resultante, define los servicios de los cuales estará compuesto el sistema, sus interacciones, y con qué tecnologías serán implementados. Las interfaces que utiliza cada servicio para exponer su funcionalidad son gobernadas por contratos, que definen claramente el conjunto de mensajes soportados, su contenido y las políticas aplicables.

Se han desarrollado 3 grandes servicios, como pueden verse en la figura 1, que a continuación se detallan:

- Enterprise Framework Services: se desarrolló el Framework del Hospital Posadas con la idea de que todos los sistemas que se vayan a realizar lo utilicen. Este Framework posee las siguientes funcionalidades: seguridad, logging, acceso a datos, manejo de excepciones, controles Web, caching y configuración. Como Servicio Web se exponen los métodos de Seguridad y Logging

- Reporting Services: es una plataforma de reportes basada en servidores, la misma puede ser empleada para crear y administrar reportes tabulares, de matrices, gráficos y de libre formato, la información de estos reportes pueden provenir de diferentes orígenes de datos. Los reportes definidos pueden ser administrados y consultados a través de un sistema Web propio o por Web Services desde las aplicaciones del Hospital. Estos reportes pueden ser obtenidos en distintos formatos (HTML, Excel, Pdf, Tiff, etc.)

- GPC Services: este servicio expone la funcionalidad

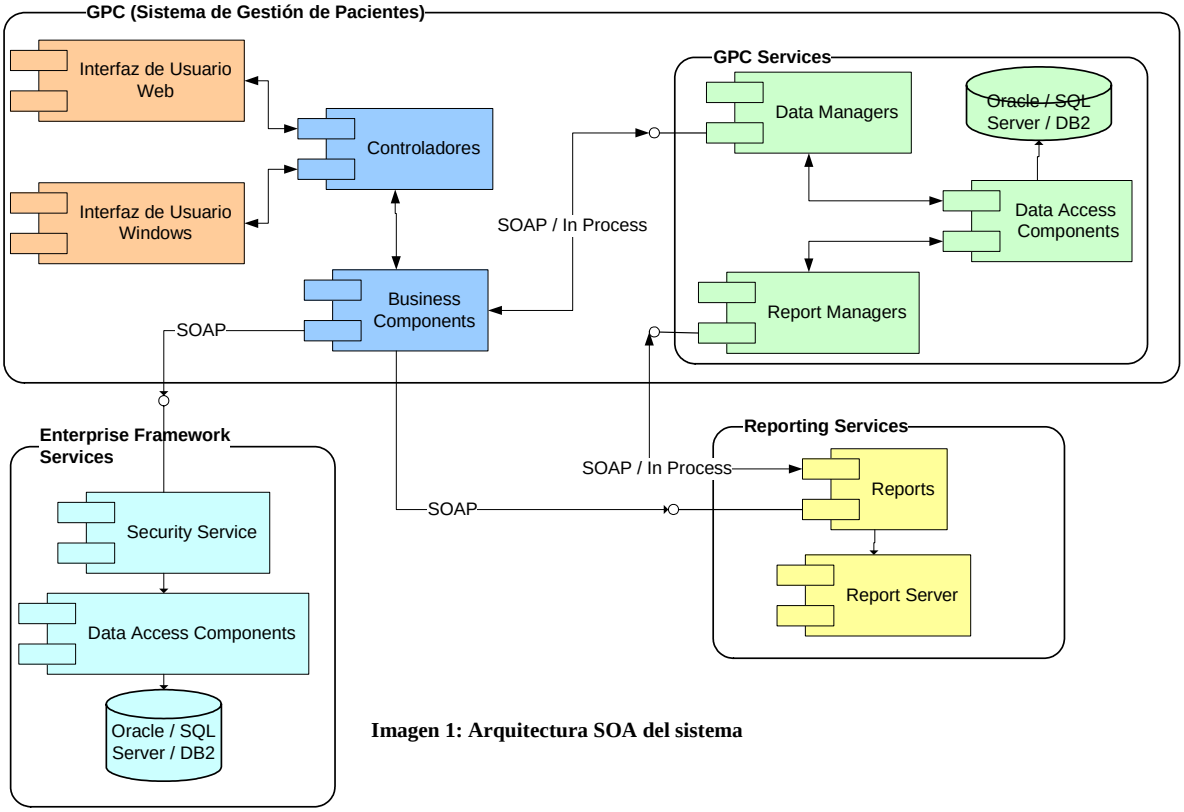

Imagen 1: Arquitectura SOA del sistema

del sistema de Gestión de Pacientes.

Dado que los tiempos de llamado no son despreciables, gracias a la comunicación de la red, tamaño de los mensajes, etc.; se ha centralizado la invocación a los servicios desde las clases de Business Components. Este componente administra la Cache de las entidades que no cambian frecuentemente y además encapsula los Proxies necesarios para invocar a los Servicios Web.

Más allá de la arquitectura seleccionada vale la pena mencionar que se ha utilizado en el sistema el Patrón de Diseño Model – View – Controller, en su versión mejorada denominada UI Process. Las principales ventajas son las siguientes:

- Abstraer el flujo de navegación de la interface: el flujo de navegación de una aplicación es una regla de negocio y, como tal, no debería estar implementada en la capa de presentación. Se trata de sacar el flujo de navegación del código en sí de la aplicación, lo que aumenta la flexibilidad y portabilidad de ésta.

- Facilita la reutilización del modelo de una aplicación en otra: permite desarrollar una aplicación Web que probablemente más adelante necesite tener un interface Windows y viceversa. Mediante la utilización del UIP modificar la interfaz de la aplicación se vuelve una tarea realmente sencilla.

3. INGENIERÍA DE USABILIDAD Y DISEÑO DE INTERFACES DE USUARIO

Uno de los principales objetivos planteados en el proyecto, tal como mencionamos, es lograr una experiencia de usuario favorable, principalmente en términos de agilidad, dada la gran cantidad de operaciones que se realizan en determinadas funciones del sistema, como por ejemplo la "Asignación de Turnos", o la "Búsqueda de Pacientes". Esto no es un tema menor, dado que en las aplicaciones Web tradicionales los usuarios interactúan mediante formularios, que al enviarse, disparan un requerimiento HTTP[iii] al servidor Web. El servidor efectúa un proceso (recopila información, procesa números, etc), y le devuelve una pagina HTML[iv] al cliente. Este proceso provoca que la interacción con el usuario se detenga cada vez que la aplicación necesita algo del servidor. Por lo tanto, se desperdicia mucho ancho de banda, ya que gran parte del texto HTML enviado en la segunda página Web, ya estaba presente en la primera.

Para solucionar este problema se ha implementado una de las últimas tecnologías en desarrollo Web, denominada AJAX[v]. Esta técnica permite crear aplicaciones interactivas mediante la combinación de tres tecnologías ya existentes:

- HTML y Hojas de Estilo para presentar la información.
- DOM[vi] y JavaScript, para interactuar dinámicamente con los datos.
- XML y XSLT, para intercambiar y manipular datos de manera asincrónica con un servidor Web (aunque las aplicaciones AJAX pueden usar otro tipo de tecnologías, incluyendo texto plano, para realizar esta labor).

El concepto es cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background (por detrás), los datos necesarios para actualizar la página, pero renderizando nuevamente la misma solo en las "porciones" necesarias. La página no se recarga nuevamente por completa

En la imagen 2 se muestra un ejemplo de la utilización de esta tecnología en la aplicación:

Imagen 2: Control AJAX

Este control "sugiere" dinámicamente los apellidos que coinciden con el texto que ingresa el usuario. Esto reduce notablemente el tiempo de búsqueda comparado con una aplicación Web tradicional, donde habría que en principio, ingresar el criterio, realizar la búsqueda, seleccionar la opción deseada, y cada una de estas interacciones con una recarga tediosa de la página.

En el diseño de la interfaz también se han considerado y aplicado heurísticas de la Ingeniería de Usabilidad, como la utilización de accesos rápidos, coherencia y consistencia de fuentes y colores, ayuda en línea contextual, prevención de errores y mensajes claros y amigables.

4. TECNOLOGÍAS UTILIZADAS

A continuación se detallan las tecnologías que fueron utilizadas tanto para el desarrollo como para la organización y el diseño del sistema.

- Proceso Unificado (UP)[2]: es un proceso iterativo para proyectos que utilizan Análisis y Diseño Orientado a Objetos. Ha sido adoptado ampliamente gracias a que se basa en las buenas prácticas del proceso de desarrollo de software como ser el ciclo de vida iterativo, desarrollo dirigido por el riesgo, descripción consistente

- UML[2]: el lenguaje unificado de modelado es un lenguaje para especificar, visualizar, construir y documentar los artefactos de los sistemas software, así como para el modelado del negocio y otros sistemas no software. Es un requisito esencial cuando se utiliza UP.

- Repositorio de Archivos: CVS[8] como Servidor y Tortoise[9] como cliente.

- Herramientas de Calidad (QA): revisión de código (ReSharper[12] y DevPartner[13]).

- Testing: documentos de test funcional, test de Stress y test unitario. Para la administración de incidencias se publicó un sistema Web denominado Gemini[7].

- Plataforma de Desarrollo Microsoft .NET[4]: es una plataforma de software que conecta información, sistemas, personas y dispositivos. La plataforma .NET conecta una gran variedad de tecnologías de uso personal y de negocios, de teléfonos celulares a servidores corporativos, permitiendo el acceso a información importante, donde y cuando se necesiten. Para el desarrollo se ha utilizado el lenguaje C#, ASP.NET y XML Web Services[3].

Imagen 3: Plataforma .NET

- Base de Datos: SQL Server 2000 y además el sistema de reportes Reporting Services.

- Interfaz de Usuario Web: AJAX y Controles Web Personalizados. Además el sistema está preparado para ser multi-idioma ya que utiliza archivos de recurso como repositorio de texto.

- Estimaciones[15]: COCOMO II[16] es un modelo que permite realizar estimaciones y planificaciones de proyectos de sistemas de información. Pertenece a la categoría de modelos de estimación basados en estimaciones matemáticas. Está orientado a la magnitud del producto final, midiendo el tamaño del proyecto en líneas de código principalmente.

5. CONCLUSIÓN

Las investigaciones realizadas durante el proyecto y el desarrollo propiamente dicho nos han permitido conocer e implementar las últimas tecnologías en desarrollo de software, posibilitando que cada integrante del equipo busque su espacio, generando un ambiente de coparticipación general. En virtud de lo mencionado podemos afirmar que hemos obtenido resultados muy favorables ampliando nuestros conocimientos en las siguientes áreas:

- Arquitectura de Software: Desarrollo de un sistema sólido y escalable para futuros desarrollos en el hospital.

- Interfaz de Usuario: Aplicación de principios y patrones de diseño de Interfaces para usuario.

- Administración y gestión: Análisis de técnicas de estimación, desarrollo de herramientas para aplicar estos conceptos.

- Pruebas y Calidad de Software: Análisis de distintos tipos de prueba y estándares de desarrollo aplicables al proyecto, implementación de herramientas de seguimiento y administración de errores.

Además del aprendizaje, es aun más gratificante para cada uno de nosotros el poder aportar a la sociedad un bien tangible, con los cimientos necesarios para que futuros alumnos puedan continuar desde diferentes áreas con este proyecto.

6. BIBLIOGRAFÍA

[1] Designing the User Interface - Ben Shneiderman - 3° Edition – Addison Wesley.
[2] UML y Patrones – Craig Larman - 2° Edition – Prentice Hall.
[3] Microsoft .Net Xml Web Services – Robert Tabor – SAMS.
[4] Microsoft .Net – http://www.microsoft.com/latam/net/ - Microsoft
[5] Arquitectura Orientada a Servicios – http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art143.asp - Microsoft
[6] AJAX – http://es.wikipedia.org/wiki/AJAX - Wikipedia, http://www.adaptivepath.com/publications/essays/archives/000385.php
[7] Gemini – http://www.countersoft.com/ - CounterSoft
[8] . CVSnt – http://www.cvsnt.org/
[9] . Tortoise – http://tortoisesvn.tigris.org/
[10] Microsoft.NET – www.microsoft.com/net/ - Microsoft
[11] Java - http://java.sun.com/ - Sun
[12] Resharper - http://www.jetbrains.com/index.html - Jetbrains
[13] DevPartner - http://www.compuware.com/products/devpartner/default.htm - Compuware
[14] UIP - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/uipab.asp - Microsoft
[15] Estimación del Esfuerzo Basada en Casos de Uso - Mario Peralta - Centro de Ingeniería del Software e Ingeniería del Conocimiento (CAPIS)
[16] Cocomo II - Model Definition Manual

7. REFERENCIAS

[i] SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

ii WSDL son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web. La versión 1.1 está en estado de "propuesta de recomendación" por parte del W3C. WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

iii El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW). El hipertexto es el contenido de las páginas Web, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceder a una página Web, y la respuesta de esa Web, remitiendo la información que se verá en pantalla. También sirve el protocolo para enviar información adicional en ambos sentidos, como formularios con mensajes y otros similares.

iv El HTML, acrónimo inglés de Hypertext Markup Language (lenguaje de etiquetado de documentos hipertextual), es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web. Gracias a Internet y a los navegadores del tipo Internet Explorer,

Opera, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos.

[v] Asynchronous JavaScript And XML (en inglés «JavaScript y XML asíncronos»).

[vi] El Modelo de Objetos de Documento (en inglés, Document Object Model o DOM) es una forma de representar los elementos de un documento estructurado (tal como una página Web HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. El custodio del DOM es el World Wide Web Consortium (W3C). En efecto, el DOM es una API para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (Javascript).