Uso de Patrones Arquitectónicos Web para el Diseño de una Aplicación Domótica

T. **Ballari;** H. **Molina;** E. **Wainerman;** L. **Olsina**
GIDIS - Dpto. de Computación, Facultad de Ingeniería, UNLPam
Calle 9 y 110 – (6360) Gral. Pico, La Pampa
TE +54 (0)2302 430497 Interno 6501
E-mail olsinal@ing.unlpam.edu.ar

1. Introducción

En los últimos años los sitios Web han ido ganando en interacción y funcionalidad para pasar de ser no sólo una forma de presentar información y contenido sino también aplicaciones con soporte a complejidad de software tradicional. Así, muchas aplicaciones están siendo portadas a la Web, aprovechando las características de uniformidad, ubicuidad y extensibilidad que este medio presenta.

En este contexto, el dominio de aplicaciones de Domótica se está viendo muy favorecida por el medio Web, entendiendo por Domótica –palabra surgida de doméstico e informática- como la disciplina que estudia el desarrollo de infraestructuras inteligentes en casas y edificios, como así también las tecnologías de información para soportarlas [8].

Nuestra línea de investigación se centró en una primer etapa, en el diseño y construcción de una aplicación distribuida Web para el control y monitoreo de módulos domóticos, a saber: los *subsistemas de Iluminación y de Cochera* [7, 3]. En esta segunda etapa, estamos integrando arquitecturas de software y hardware, centrada esta última en el mecanismo CAN (Controlled Area Network) [6]. El objetivo de la aplicación fue el de controlar mediante la Web el funcionamiento de los módulos de iluminación y cochera de un edificio (maqueta), con dispositivos de sensado y actuación. La aplicación, por ejemplo el módulo de cochera, mantiene el control del estado de los dispositivos físicos intervinientes, y brinda la posibilidad de acceder al edificio con la debida autenticación del usuario. Desde el punto de vista arquitectural, se decidió por una arquitectura de software de tres capas, bajo el modelo cliente-servidor, con aspectos específicos para controlar y monitorear los dispositivos y eventos en tiempo real (esto es, soft real-time). Es de especial importancia el requerimiento de actualización del estado de los dispositivos en los clientes, ante los eventos ocurridos en el modelo físico, que es realizado por el servidor mediante el mecanismo denominado callback.

Por una parte, uno de los objetivos del proyecto fue la elección de una plataforma flexible y genérica para el desarrollo de aplicaciones centradas en la Web, evaluando las distintas arquitecturas e implementaciones posibles. Una meta específica fue el crear componentes reusables basándonos para su construcción en distintos patrones arquitecturales y de diseño. Por otra parte, es necesario contar con un proceso de desarrollo sistemático y bien definido, para poder guiar al desarrollador en la distintas fases y en el proceso de documentación, el cual proceso debe ser lo suficientemente flexible como para que se adapte a la naturaleza del proyecto. Para tal fin utilizamos el proceso RUP (Rational Unified Process) [5].

En el resto de esta comunicación, nos concentraremos en la visión arquitectural, principalmente en los patrones arquitectónicos y en los principios del desarrollo.

2. Visión Arquitectural y de Desarrollo centrado en la Web

El diseño de la aplicación se centró en un modelo cliente-servidor basado en un enfoque de arquitectura de software de tres capas. Básicamente, este modelo propone tener al cliente como una de sus capas (capa de presentación) , el cual se comunica con una capa intermedia que es la del servidor (capa de lógica de negocio), siendo éste último el encargado de la manipulación de los datos con la tercera capa (capa de persistencia). Esta distribución permite que varios clientes interactuen con un servidor (o más de uno), el cual accederá a una BD (o varias) a efectos de responder a los requerimientos de los clientes. Es también la capa del servidor quien se encarga de la lógica de

negocio y comunicación con los dispositivos físicos. La selección de la arquitectura y ambientes de desarrollo como aspectos de implementación fueron discutidos en [3].

Una aplicación web (a diferencia de un sitio meramente estático), está compuesta como mínimo por tres componentes: un navegador web, un servidor web y un servidor de aplicaciones, pudiendo intervenir otros componentes, como el de persistencia, entre otros. Por lo que se puede decir que una aplicación web es un caso particular del modelo de tres capas antes citado. En este contexto, se han identificado tres patrones arquitectónicos de gran utilidad para el proceso de diseño arquitectural. Los siguientes patrones recientemente documentados [2], *Thin Web Client* (o Cliente Web Ligero), *Thick Web Client* (o Cliente Web Pesado) y el patrón arquitectural *Web Delivery* (o Distribución Web), se diferencian tanto por el nivel de requerimientos de tecnologías necesarias en las capas cliente, servidora y datos como en el modo de distribución de la funcionalidad que se puede considerar esencialmente en las dos primeras capas. Debido a las características de nuestra aplicación (ver [3]), de estos tres patrones hemos reusado y adaptado al patrón *Cliente Web Ligero* y el de *Distribución Web*.

Como ha indicado Alexander [1], "*cada patrón describe un problema que ocurre una y otra vez... y luego describe el núcleo de [una] solución al problema, de forma que se puede usar esta solución un millón de veces más, sin hacerlo de la misma forma dos veces*." Un patrón es una solución general a un problema recurrente en un contexto dado, que se lo identifica básicamente con un nombre, un planteo del problema, su solución y sus consecuencias. Los patrones arquitectónicos abstraen los principales componentes de una arquitectura de sistema para ofrecer una solución a un problema. Por lo tanto, aplicar patrones arquitecturales permite al diseñador concentrarse en los detalles propios de su aplicación, utilizando soluciones ya existentes a los problemas recurrentes dentro de un dominio dado. También permite que su diseño sea más fácil de comprender por otros participantes del proceso al utilizar una estructura bien identificada, facilitando las tareas de documentación, comunicación y potenciales extensiones o adaptaciones del sistema a nuevos requerimientos.

En cuanto al diseño de la aplicación Domótica, para contemplar las dos funcionalidades principales como son los módulos de administración y mantenimiento de usuarios, y los módulos de control y monitoreo de funcionalidad de iluminación y cochera, tomamos la decisión de utilizar los patrones *Cliente Web Ligero* para la primera y *Distribución Web* para la segunda.

El patrón *Cliente Web Ligero* se usa en el caso en que no hay funcionalidad en la capa cliente, la cual sólo requiere de un navegador estándar. Toda la lógica de la aplicación se ejecuta en la capa del servidor.

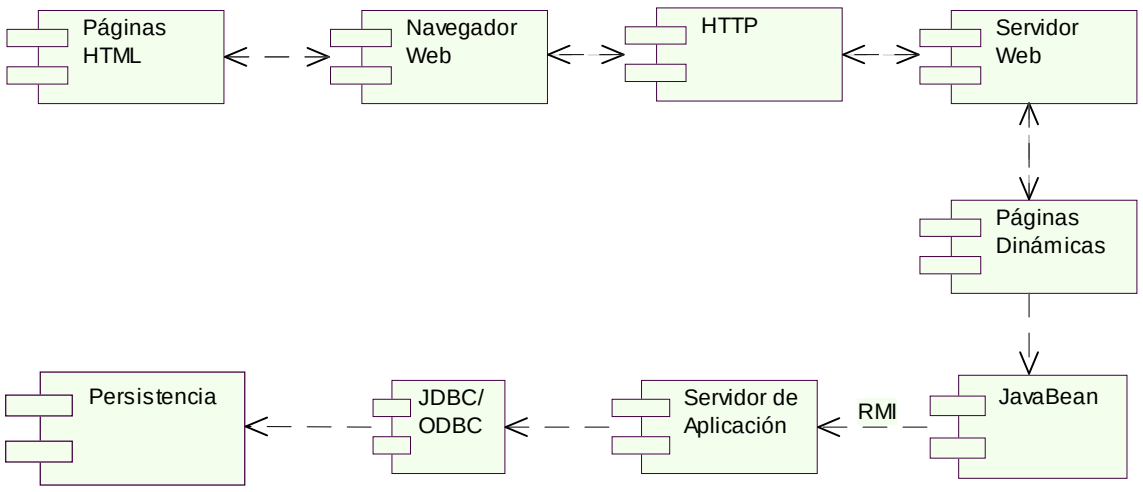

Fig. 1. *Arquitectura de la aplicación para manejo de usuarios siguiendo el patrón Thin Web Client*.

Como se ha mencionado anteriormente, la administración de los usuarios fue implementada mediante este patrón. En este caso, el navegador le presenta al usuario una interface basada en HTML, mediante la cual se pueden llenar formularios, presionar botones para enviar los datos ingresados al servidor, etc. Como se observa en la figura 1, el servidor web, recibe las peticiones del cliente a través

del protocolo HTTP, entrega estas peticiones al módulo que genera las páginas dinámicas, y éste ejecuta el código embebido en HTML, que se comunica con el servidor de aplicación a través de un componente JavaBean. El servidor de aplicación, para realizar la autenticación de los usuarios, o la modificación de los datos de éstos, o para revisar el registro de eventos, accederá al componente de persistencia a través del puente JDBC/ODBC.

En el patrón arquitectónico *Distribución Web* la comunicación entre cliente y servidor no se realiza únicamente a través del componente HTTP, sino que entra en juego, como por ejemplo en nuestro caso, el uso del componente RMI, que brinda el soporte para la comunicación de objetos distribuidos en el entorno Java, como se aprecia en la figura 2. En esta instancia, el navegador se limita a ser un contenedor de los objetos del cliente, ejecutándolos dentro de su máquina virtual Java, y sólo interactúa con la aplicación al momento de cargar, iniciar y detener su ejecución. Una vez que se ejecuta la aplicación cliente, esta interactúa con el servidor de aplicación usando el protocolo RMI (u otros), mediante el cual puede invocar métodos de un objeto remoto exportado por el servidor. Este, a su vez, puede invocar a métodos remotos sobre el cliente mediante RMI para actualizarlo y enviarle los datos de cambios ocurridos en el fenómeno físico (mecanismo callback).

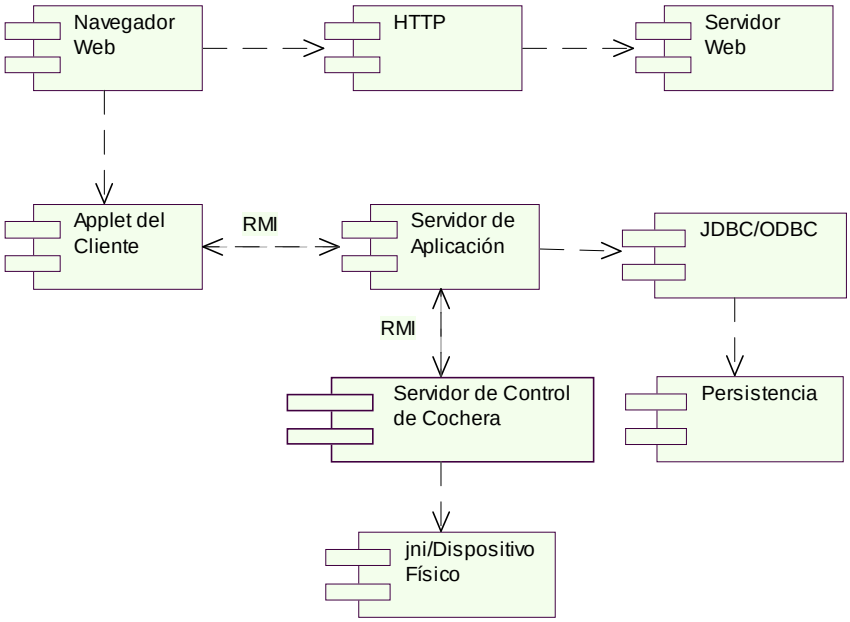

Fig. 2. *Arquitectura de la aplicación para manejo de usuarios siguiendo el patrón Thin Web Client*.

El servidor, ante los mensajes enviados por los clientes, y contemplando la lógica de la aplicación, puede modificar el estado de los dispositivos o recibir información de los cambios. Cabe destacar que el servidor de cochera (o iluminación) accede a los dispositivos físicos a través del componente JNI que encapsula las funciones de la placa adquisidora de datos (o a las funciones de interface con la arquitectura CAN, según nuestra línea actual de trabajo).

3. Consideraciones Finales

Se ha seleccionado para el desarrollo de la aplicación Domótica distribuida un enfoque basado en tecnologías Web. Esto hizo que el resultado obtenido fuese una aplicación fácilmente accesible a través de una interface bien conocida y aceptada cada vez más entre los usuarios. Es importante destacar que la investigación realizada en este campo, nos permitió reusar y personalizar mecanismos de gran utilidad como son los patrones arquitectónicos Web. Mediante el empleo de patrones, nos fue posible resolver problemas de diseño arquitectural de la aplicación, de un modo flexible y que a su vez fueran adaptables a otros dominios.

Otro aspecto importante en el desarrollo de nuestra aplicación fue la selección de un proceso sistemático y bien definido para las distintas fases, que además favoreciera al proceso de documentación. Nos estamos refiriendo al proceso RUP (Rational Unified Process), cuya estrategia de

proceso de desarrollo es una dirigida por *Casos de Uso*, centrada en la *Arquitectura*, y que desde el punto de vista dinámico es *iterativa e incremental* [5].

Agradecimientos

Esta investigación es soportada por el proyecto UNLPam-09/F014. Agradecemos la participación de Giles, A; Miguel, F., y Lapine, P., tanto en el diseño como en la construcción de la maqueta del edificio y su circuitería.

Referencias

1. Alexander, Christopher (1977), *A Pattern Language*, Oxford University Press.
2. Conallen, J., (1999), *Building Web Applications with UML*, Addison-Wesley.
3. Echeverría, E.; T. Ballari; H. Molina; E. Wainerman; L. Olsina, (2000), *Arquitectura Centrada en la Web para el Control y Monitoreo de Funcionalidad Domótica*; Proceedings del VI Congreso Argentino de Ciencias de la Computación. (CACIC), Ushuaia, Arg.
4. Jong Jin Kim, Jin Kim Jong (1998), *Intelligent Buildings*, Butterworth-Heinemann Publisher.
5. Kruchten, Philippe (2000*), The Rational Unified Process, An Introduction*, Addison-Wesley.
6. Lawrenz, W. (1997), CAN *System Engineering: Form Theory to Practical Applications*, Springer Verlag; ISBN: 0387949399
7. Olsina, L., Echeverría, E., Miguel, F., Giles, A. (2000), *Domotic Monitoring by using the Web*, Proceed. of AADECA 2000, Asociación Argentina de Control Automático, Bs. As. pp. 189-194
8. Quinteiro González, J., Lamas Graziani, J.; Sandoval, J., (1999), *Sistemas de Control para Viviendas y Edificios: Domótica*, Ed. Paraninfo, Mad. Spain.