GENERACIÓN AUTOMÁTICA DE ITINERARIOS CULTURALES APLICANDO EJB Y PROGRAMACIÓN CON RESTRICCIONES

J.A. Ortega, F.Ruz, M.J. Escalona, M. Mejías, J.M. Márquez

Dpto. de Lenguajes y Sistemas Informáticos
Facultad de Informática y Estadística
Universidad de Sevilla
ortega@lsi.us.es

La aparición cada vez más frecuente de diferentes dispositivos fijos y sobre todo móviles hace que la arquitecturas de las aplicaciones tengan que separar y definir con más precisión las distintas capas qu componen la misma, de forma que podamos diferenciar en la aplicación la capa de lógica de negocio y l capa de presentación. Sólo de esta forma podremos crear un sistema escalable, modular y flexible, qu pueda ser accedido por dichos dispositivos (front-ends) sin cambiar la lógica de negocio del sistema Además es deseable que esta lógica pueda ser reutilizada otros proyectos de características similares.

La arquitectura J2EE, proporciona un excelente framework para arquitecturas distribuidas que usa e la capa de negocios componentes EJBs. El uso de estos componentes junto con la programación co restricciones en Java es utilizado en este artículo para modelar un sistema de generación automática d itinerarios culturales [2][10].

1.Introducción

El Instituto Andaluz de Patrimonio Histórico (IAPH)[3] es el organismo de la Junta de Andalucí encargado de gestionar la información de los bienes patrimoniales de la comunidad andaluza. Est información puede ser una fuente inagotable de consultas tanto para personal especializad (investigadores y catalogadores del patrimonio) como para personal no especializado (ciudadanos qu deseen acceder al patrimonio histórico de su localidad). En concreto se pretende modelar un sistema par que los usuarios puedan obtener itinerarios culturales en base a unas determinadas restricciones impuesta por ellos mismos.

Hasta ahora, para obtener cualquier tipo de información del IAPH había que solicitar la misma bie por teléfono o bien vía mail. Este procedimiento indudablemente es lento y es dependiente de l disponibilidad y capacidad del IAPH, que en ocasiones no puede enfrentarse a todas las solicitudes a mismo tiempo, provocando esto un retraso en las respuestas a las peticiones.

En [2][10] se plantea la necesidad de desarrollar una nueva metodología capaz de adaptar el actua sistema de información del IAPH a las nuevas necesidades y requisitos que el avance de la tecnología, en especial Internet, está generando. De esta forma se desarrolla un sistema de trabajo capaz de mejorar e procedimiento existente en la actualidad en el IAPH, que pueda ser accedido por cualquier persona, e cualquier momento y que genere las respuestas a las peticiones en tiempo real con idea.

Mediante el uso de EJBs se pretende aprovechar todas las ventajas que nos ofrecen estos componente middleware y que hacen posible la separación de las capas de la aplicación. Sólo mediante una separació clara de dichas capas conseguiremos un sistema flexible, modular y fácilmente ampliable, haciend posible reutilizar la lógica de negocio en diferentes front-ends.

La programación con restricciones también será utilizada para permitir al usuario la realización d consultas más flexibles y potentes con el fin de generar el itinerario más acorde con sus preferencias objetivo principal del sistema.

Aunque la resolución de restricciones mediante Java no es un paradigma muy extendido actualmente (ni siquiera Sun ofrece una API), se han desarrollado o están desarrollándose diversas herramientas qu son capaces de resolver restricciones mediante este lenguaje, tal y como JCK [6], JCL[7] o JSOLVER[4].

El presente artículo plantea resolver este sistema utilizando la plataforma J2EE[5] y en especial los EJB[1] para crear la arquitectura que proporcione este soporte. Este sistema será referido como Sistema Generación Itinerarios Culturales EJB o *sgicEJB* a lo largo de todo el artículo.

Para ello se muestra, en el apartado 2, un resumen acerca de los EJB y de las ventajas que suponen su uso para la generación automática de itinerarios culturales. En el apartado 3 se presenta la metodología de trabajo para la generación automática de itinerarios culturales y la forma en la que puede plantearse esta mediante el uso de EJBs.

2.EJB

EJB es una especificación que intenta simplificar al máximo las tareas del programador a la hora de desarrollar componentes para una arquitectura distribuida. Esto quiere decir que *el programador sólo se ha de preocupar por la lógica de negocio de la aplicación*, ya que los servicios de bajo nivel tales como las transacciones, concurrencia, gestión de recursos, la persistencia, la seguridad o las conexiones con la base de datos son suministrados de forma automática por los servidores EJB (y más concretamente por los contenedores EJB). Los componentes EJB no son más que clases Java que representan la lógica de negocio del componente y que residen dentro del contenedor. De esta forma se podrán crear de manera sencilla aplicaciones distribuidas en el lado del servidor que sean escalables, transaccionales, multiusuario y seguras. Además, la perfecta separación que J2EE hace de las capas que componen su arquitectura permite que estas, y más concretamente la capa de negocios, puedan ser reutilizadas.

Esta facilidad proporcionada por los EJB será aprovechada para modelar el sistema que permita generar itinerarios culturales de forma automática

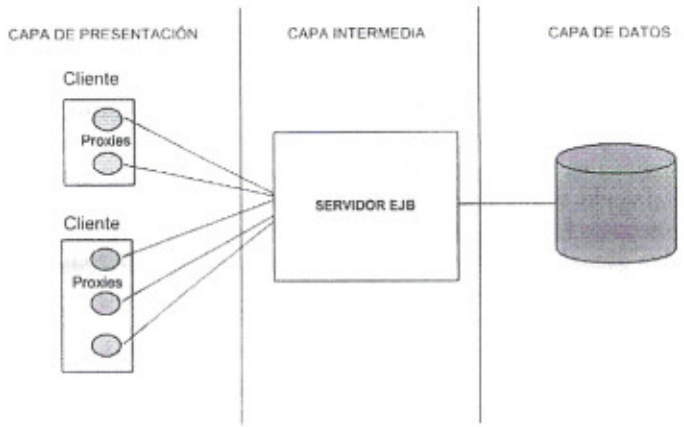

Figura. 1. Arquitectura EJB

Como hemos mencionado, uno de los objetivos principales de la tecnología EJB es la de ser una arquitectura de componentes standard para la construcción de sistemas de negocios distribuidos y orientados a objetos. Este es un objetivo un poco ambiguo que no podría ser llevado a cabo por un único modelo de componentes. Es por ello que la especificación define tres tipos diferentes de componentes EJB:

a) Beans de sesión.

Modelan los procesos de negocio de la aplicación, como por ejemplo el cálculo de las restricciones introducidas por el usuario. Dado que el cliente solicitará un servicio de uno de estos beans, cada cliente tendrá su propia instancia del bean; dichas instancias no pueden ser compartidas entre múltiples clientes. Los beans de sesión pueden dividirse en dos tipos: con estado y sin estado.

- *Sin estado:* no almacenan información alguna acerca del cliente entre llamadas
-*Con estado:* almacenan información entre llamadas de un cliente.

Para el desarrollador, es importante saber que los bean de sesión no sobreviven a una caída del servidor, por lo que si esto sucede el cliente deberá reestablecer un nuevo objeto de sesión.

b) Beans de entidad.

Modelan los datos de negocio de la aplicación, mapeando una clase Java con una fuente de datos. Esta puede ser una fila de una base de datos, una tabla entera o algún otro tipo de dato representado en una

base de datos. Cada bean de entidad posee una clave primaria asociada que identifica a los datos que contiene. Sería muy difícil controlar los cambios de múltiples copias de los mismos datos, por lo que sólo podrá existir en el sistema una instancia de un bean de entidad por cada clave primaria dada (esto es así incluso en sistemas distribuidos).
Los beans de entidad pueden también separarse en dos tipos: los que manejan su propia persistencia (BMP) y los que delegan esta función en al contenedor EJB (CMP).

c) Beans de mensajería.

Permiten procesar mensajes asíncronos. Actúa como un escuchador de mensajes JMS, los cuales pueden ser enviados por cualquier componente J2EE (una aplicación cliente, otro EJB, o un componente Web) o por cualquier aplicación o sistema JMS que no use la tecnología J2EE.

2.1 Ventajas del uso de EJB

El uso de EJB como componentes de lógica de negocio y datos para la creación del sistema de generación automática de itinerarios culturales supone las siguientes ventajas:

- *Escalabilidad y portabilidad* a través de diferentes servidores de aplicación, siguiendo claramente con la filosofía WORA[12] de Java.
  Sin contar con la portabilidad que ofrece intrínsecamente la plataforma Java, las aplicaciones que cumplen con la especificación EJB pueden ser escritas una vez, y después desplegadas en cualquier plataforma de servidor que soporte la especificación Enterprise JavaBeans. Estas plataformas podrán añadir funcionalidades añadidas, lo que permitirá migrar sgicEJB al servidor que se adapte mejor a las necesidades del IAPH, teniendo en cuenta factores como la seguridad, la potencia, la escalabilidad o la fiabilidad.

- *Reusabilidad* de componentes, que actúan a modo de middleware (entre el cliente y el servidor) , debido a que estos son empaquetados como un todo.
  Hoy día existen múltiples clientes en el front-end (puestos / navegadores, PDAs, Móviles, NCs, NetPCs...). Como back-end hay aún una mayor heterogeneidad en cuanto a plataformas y fuentes de datos. El middleware Java de la especificación EJB conecta ambos mundos. En otras palabras, la lógica de negocio puede implementarse como componentes reusables en la capa intermedia, proporcionando acceso a cualquier tipo de cliente al back-end y front-end.
  Podemos pensar en sgicEJB como un componente que recibe unos inputs (restricciones) y genera unos outputs (itinerario generado que mejor se adapta a esas restricciones), la cual podemos utilizar en distintos proyectos o sistemas. Este componente podría a su vez asociarse a otros componentes para generar aplicaciones mayores, siguiendo lo que algunos autores han denominado acertadamente la filosofía LEGO[9]. De esta forma, podríamos tener un componente por cada comunidad autónoma que generase los itinerarios de bienes correspondientes a las mismas. Uniendo todos los componentes obtendríamos un portal para la generación de itinerarios culturales a nivel nacional. Como puede observarse, este sistema podría ampliarse fácilmente, para ello bastaría con crear un nuevo componente y unirlo al sistema.

- *Soporte para servicios Web*. EJB 2.1 permitirá (aún está en draft) a los desarrolladores exponer beans de sesión sin estado y de mensajería como servicios Web basados en SOAP, haciéndolos de esta forma accesible a cualquier cliente SOAP 1.1 compliant. Por ejemplo, usando SOAP podríamos invocar a nuestro sgicEJB desde otras plataformas de servicio Web como .NET de Microsoft, Perl, Apache Axis, C, C++ y muchos otros lenguajes y plataformas. Las nuevas características de servicios Web en EJB 2.1 proporcionan un nivel de interoperabilidad entre plataformas nunca visto hasta el momento. Ello nos permitiría Front-Ends desarrollador con diferentes lenguajes.

- *Incremento de la productividad de los desarrolladores*, ya que los mismos sólo han de preocuparse por la lógica de negocio de la aplicación (los EJB harán el resto).
  El desarrollo del sgicEJB se reduciría básicamente a la creación y resolución de restricciones en función a los datos introducidos por los usuarios.

3.Generación de itinerarios

La expresividad que el conocimiento cualitativo puede ofrecer a la hora de expresar una consulta es aprovechada para su aplicación en la generación de itinerarios culturales[2][10].

Una de las principales ventajas que el usuario encuentra cuando utiliza el razonamiento cualitativo es que puede hacer uso de toda la potencia y flexibilidad de expresión que este ofrece. De esta forma es posible generar restricciones cualitativas que serán consideradas a la hora de generar el itinerario más adecuado.

Aún así, el usuario podrá seguir haciendo uso de datos discretos que le ayudarán a definir mejor el itinerario deseado por el mismo. En definitiva, se debe permitir que un usuario obtenga un itinerario en base a unas determinadas necesidades o deseos (restricciones), los cuales podrán ser expresados de forma tanto cualitativa como cuantitativa.

Por ejemplo, un usuario podría consultar el mejor itinerario para que, partiendo de Sevilla, visitar el mayor número de bienes del tipo "cofre" pertenecientes a la edad media. Además, el usuario puede indicar que el coste total de la visita no sea superior a los 20 euros y que la distancia máxima desde el punto de partida (en este caso, Sevilla) sea aproximadamente igual a 200 kilómetros.

Como resultado de la investigación, se presentó un algoritmo de trabajo que resumía el proceso que ha de llevarse a cabo para conseguir generar un itinerario a partir de las restricciones impuestas por el usuario.

3.1 Generación de itinerarios con EJB

El algoritmo de trabajo para la generación automática de itinerarios culturales mencionado antes podría resumirse en tres pasos. A continuación intentaremos profundizar un poco más en estos pasos y explicaremos cómo modelar el sistema utilizando la plataforma Internet mediante el uso de una arquitectura J2EE y más concretamente el uso de EJBs.

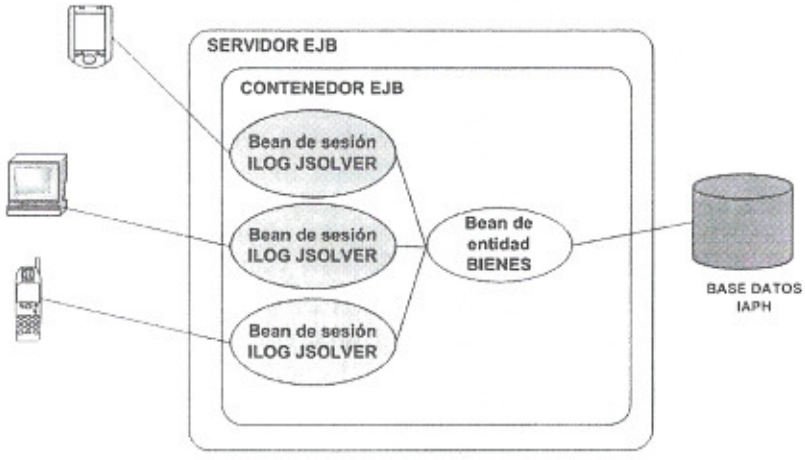

Figura 2. EJB aplicado a la generación automática de itinerarios culturales

1. - Introducción por parte del usuario de las restricciones que se han de tener en cuenta a la hora de generar el itinerario. Estas restricciones podrán ser tanto cualitativas (precio no superior a 10euros) como cuantitativas (que se encuentren en la provincia de Sevilla). La pantalla que recogerá dicha información también dará la posibilidad al usuario de elegir entre diferentes tipos de optimización del itinerario (espacio, tiempo, precio).

Esta interface gráfica deberá estar realizada principalmente en HTML, si bien es posible el uso de JSP si es necesario que la página tenga algún tipo de comportamiento dinámico como es el caso de la información de errores presentada en el siguiente punto.

probación de la información introducida por el usuario. Si el usuario introduce valores correctos (ha introducido la restricción de precio pero no ha indicado el precio de referencia) l mismo con el mensaje de error correspondiente. Si bien esto podría hacerse utilizando el lado del cliente, es más seguro y recomendable llevar a cabo esta comprobación en el lado Para ello podríamos usar un Servlet que se encargue de hacer dichas comprobaciones, do a la página del formulario si se produce algún error. En este punto se recomienda el uso framework que implementa el patrón MVC

s introducidos por el usuario son correctos, el Servlet redireccionará la petición al bean de ondiente, que vemos en el siguiente punto.

ación de las restricciones. Una vez que se han validados los parámetros seleccionados por el que convertir los mismos en restricciones.

plificar estas ideas, se muestra en la figura 3 la pantalla principal del prototipo que e está realizando. Esta interfaz no es definitiva pero como se puede observar, recoge las s en los puntos anteriores.

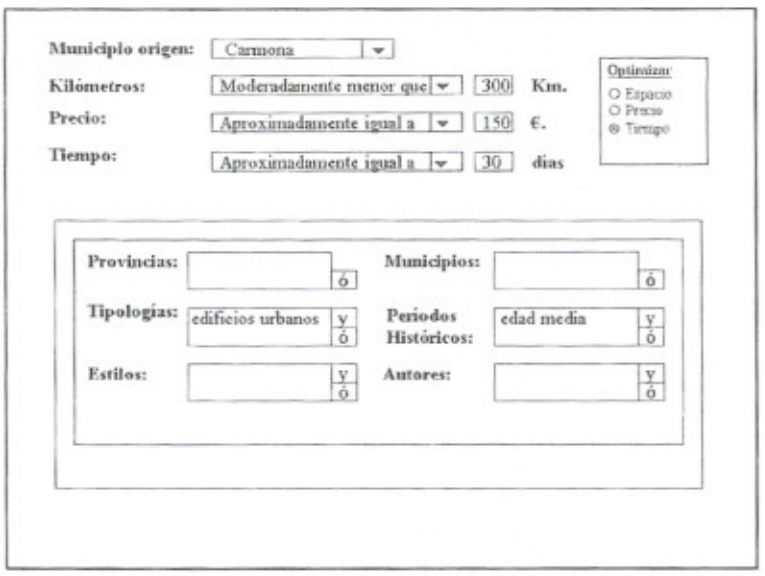

Figura 3. Prototipo de interfaz del programa de generación de itinerarios culturales

s parámetros seleccionados y/o introducidos por el usuario, se generarán una serie de e serán necesarias resolver para obtener el itinerario deseado. En la programación con puede hacer uso de un algoritmo genérico de búsqueda que resolverá las restricciones embargo, en nuestro caso hay una mezcla de restricciones continuas y concretas y seguir una metodología de trabajo diferente. Dicha metodología consiste básicamente en restricciones continuas en restricciones concretas y resolver las mismas, junto con el resto concretas introducidas por el usuario, mediante el algoritmo correspondiente.

a figura 2 aparezca un solo bean de entidad, esto es sólo una representación general de las se de datos del APH. Realmente tendremos un bean de entidad por cada tabla de interés base de datos. Estas tablas de interés serán aquellas que se encuentren en los módulos interés para el problema a resolver[2]. Estos beans de entidad serán compartidos por de sesión del sistema.

le sesión representan la lógica del sistema. Esta lógica se encargará de obtener de la base cir, de los beans de entidad) aquella información que es relevante al usuario. Para ello lver las restricciones que el usuario a introducido en la pantalla inicial. Aunque la estricciones mediante Java no es un paradigma muy extendido de momento, existen tes software como JCK [6] o JCL[7] que implementan la misma, siendo la más OG JSOLVER[4].

entidad que mapean las tablas que nos interesan del IAPH podrán ser de tipo CMP rente simplicidad de dichas tablas. De esta forma estaremos aprovechando una de las

mayores ventajas que nos ofrecen los beans de entidad. Por otro lado, el bean de sesión que modele la generación y resolución de restricciones será sin estado, ya que éste no es necesario en el transcurso de las operaciones. Los diferentes beans de sesión que aparecen en la figura 2 representan las instancias que se crean para cada cliente de dicho bean.

Opcionalmente podrían utilizarse beans de mensajería para comunicar las preferencias de los usuarios a la hora de hacer las consultas, con vistas a realizar algún tipo de tratamiento estadístico que pudiera resultar de utilidad.

4. Conclusiones y trabajos futuros

Una vez que se ha planteado la posible arquitectura del sgiEJB, la implementación del mismo no parece excesivamente compleja. Quizás la mayor complejidad se encuentre en conocer a fondo el problema y el determinar los tipos de predicados que se pueden dar, aunque este problema queda parcialmente resuelto gracias a anteriores trabajos [2][10]

Aunque la especificación EJB está orientada a la construcción de sistemas tal vez más complejos, hemos de ver sgicEJB como una apuesta de futuro, que nos permitirá ampliar el sistema fácilmente y nos facilitará la integración con distintos tipos de front-ends.

Vemos que el uso de EJBs posee numerosas ventajas que hacen que sea un paradigma adecuado para definir una arquitectura que resuelva el problema planteado.

Dada la separación de capas que se ha planteado en el sistema, no sería difícil migrar el sistema inicial (navegador de un equipo fijo como front-end) a otros tipos de front-ends quizás más adecuados.

Podríamos pensar en la adaptación de dispositivos móviles para que desde ellos también fuera posible obtener la ruta más adecuada según el lugar en el que nos encontrásemos. En este sentido, la unión de tecnologías de comunicación móviles como GPRS, UMTS, Bluetooth o Wi-FI y dispositivos tan avanzados como PDAs o móviles de última generación podrían ser usados para llevar al extremo la potencia de sgiEJB. Sería posible capturar la localización exacta de un usuario y, en base a la misma, devolverle el itinerario más adecuado al mismo. Por lo tanto, la generación de distintos front-ends que usen sgiEJB se plantea como una línea de investigación interesante.

Referencias

1. EJB:Enterprise Java Bean. http://java.sun.com/products/ejb/
2. R.M. Gasca, M.J.Escalona, J.A. Ortega, M.Mejías, J.Torres. *Aplicación de la programación con restricciones a la elaboración automática de itinerarios culturales*. Congreso de Turismo y Tecnologías de la Información y las Comunicaciones: Nuevas Tecnologías y Patrimonio. Madrid, Octubre 2001.
3. IAPH: Instituto Andaluz del Patrimonio Histórico. http://www.iaph.junta-andalucia.es/
4. ILOG JSOLVER. http://www.ilog.com/products/jsolver/
5. J2EE: Java 2 Enterprise Edition. http://java.sun.com/j2ee/
6. JCK: Java Constraint Kit. http://www.pms.informatik.uni-muenchen.de/software/jack/
7. JCL: Java Constraint Library. http://liawww.epfl.ch/~torrens/JCL/
8. JSP : Java Server Pages. http://java.sun.com/products/jsp/
9. R. Monson-Haefel : *Enterprise JavaBeans*. Oreilly & Associates, Junio1999
10. J.A Ortega, M.J Escalona, J. Torres, R.M Gasca, M.Mejías: Generación automática de itinerarios culturales definidos semicualitativamente utilizando técnicas de satisfacción de restricciones. JARCA 2002
11. Struts. http://jakarta.apache.org/struts/index.html
12. WORA. http://java.sun.com/features/1997/aug/wora.html