Gestión de una biblioteca utilizando servicios Web

Mª Nieves Carralero Colmenar

Profesora de Educación Secundaria
IES Pedro Mercedes. Cuenca

En este artículo se muestra el desarrollo de una aplicación para la gestión de una biblioteca utilizando servicios web. El desarrollo se ha realizado con un servidor de aplicación Glassfish que atiende servicios Java. Los clientes, por su parte, se han desarrollado en Java y en .Net.

Descripción de Glassfish.

Según Wikipedia, el servidor de aplicaciones Glassfish se describe como:

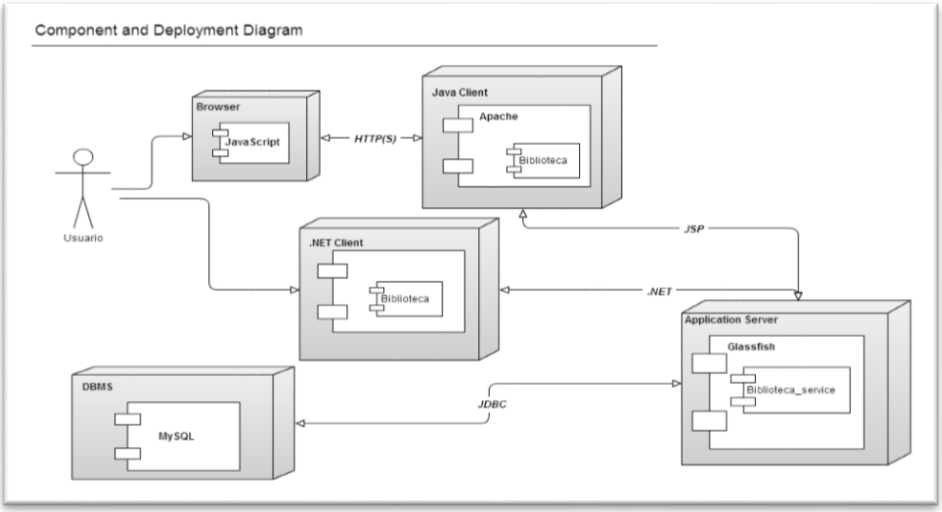
*"**GlassFish** es un servidor de aplicaciones de software libre desarrollado por Sun Microsystems, compañía adquirida por Oracle Corporation, que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Es gratuito, decódigo libre y se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL. La versión comercial es denominada Oracle GlassFish Enterprise Server (antes Sun GlassFish Enterprise Server).*

GlassFish está basado en el código fuente donado por Sun y Oracle Corporation; este último proporcionó el módulo de persistencia TopLink. GlassFish tiene como base al servidor Sun Java System Application Server de Oracle Corporation, un derivado de Apache Tomcat, y que usa un componente adicional llamado Grizzly que usa Java NIO para escalabilidad y velocidad."

Desarrollo de la aplicación.

La gestión de una biblioteca mostrada en este artículo es un desarrollo incremental con funcionalidad básica. Eso permite aislar la complejidad de la funcionalidad de la complejidad del desarrollo de servicios web.

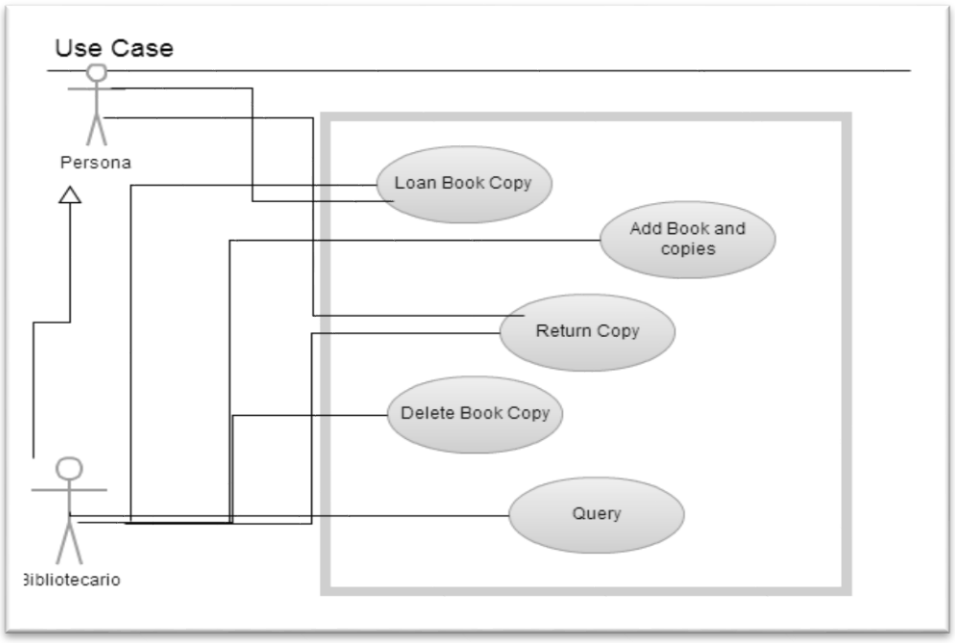
1. DIAGRAMA DE DESPLIEGUE



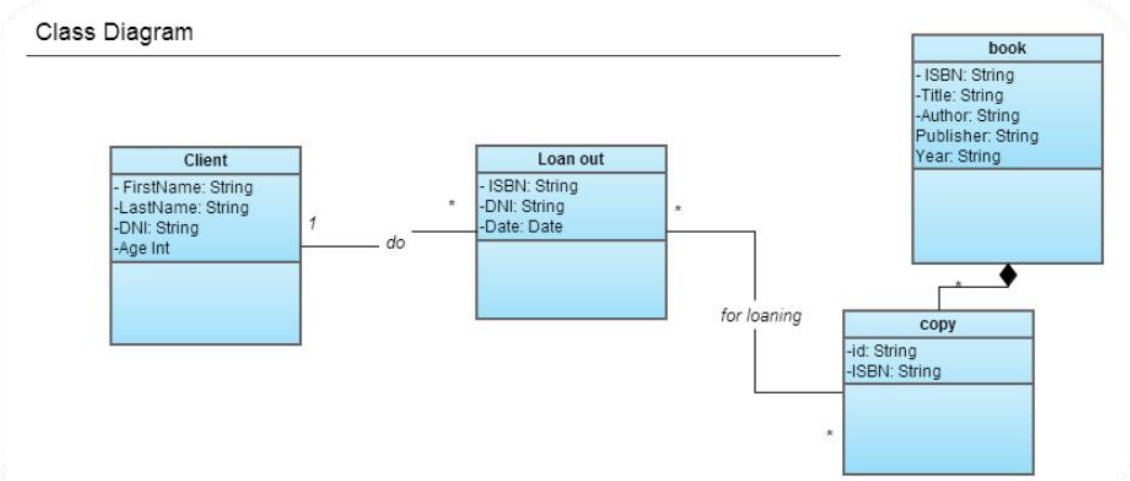
El servicio Web se encuentra en el componente de servicio aplicación que se ejecuta sobre Glassfish.
El servicio Web tiene acceso a los datos de persistencia utilizando JDBC.
El cliente Java accede al servicio Web usando JSP.
El cliente. NET accede al servicio Web utilizando. NET.
El cliente Java es una aplicación web.
El cliente. NET es una aplicación de escritorio (Standalone).

2. ESQUEMA DE CASOS DE USO

Mi aplicación tiene cinco casos de uso:
Copia de libros del préstamo: préstamos bibliotecario un libro (copia) al cliente.
Devolver copia: Cliente devuelve una copia de la biblioteca.
Añadir libro y las copias: Bibliotecario agrega un libro o una copia de la base de datos.
Eliminar Copia libro: Bibliotecario quita un libro o una copia de la base de datos.
Préstamo: Bibliotecario presta libros a un cliente.



3. DIAGRAMA DE CLASES



Client representa a la persona que pide prestado un libro (copia).
Book es una entidad abstracta.
Un libro puede tener varias copias de la biblioteca que son capaces de ser prestado a los clientes.
Préstamo a cabo establece la relación entre los clientes y los libros en una fecha determinada.

4. SERVICIOS WEB

@WebMethod(operationName = "BorrarCopia“)

public String BorrarCopia(@WebParam(name = "isbncopia") String isbncopia

« **BorrarCopia**» elimina la copia de un libro cuyo id es dado como paráme-tro. El id es el ISBN del libro original + (número de la copia)

@WebMethod(operationName = "BorrarPrestamo")

public String BorrarPrestamo(@WebParam(name = "isbncopia") String isbncopia,@WebParam(name = "nifcopia") String nifcopia) {

« **BorrarPrestamo**» elimina del préstamo fuera de la mesa de entrada cuyo nif y la identificación se dan como parámetros.

@WebMethod(operationName = "ConsultaISBN")

public java.util.ArrayList<Copia> ConsultaISBN(@WebParam(name = "copia") String copia) {

« **ConsultaISBN** » Devuelve una lista de matrices de objetos cuya copia isbn patern se le ha dado como parámetro.

@WebMethod(operationName = "AnadirLibro")

public String AnadirLibro(@WebParam(name = "copia") Copia libro, @WebParam(name = "ncopias") int ncopias) {

« **AñadirLibro** » Agrega un objeto libro y el número de copias de este libro dado como parámetro. (Primera parte de los clientes (. Net o java) se creó la copia de un objeto a través de un formulario y, a continuación, llamar a este método para agregar cada nuevo objeto).

@WebMethod(operationName = "AnadirPrestamo")

public String AnadirPrestamo(@WebParam(name = "copia") String ID_copy, @WebParam(name = "cliente") String NIF_cliente) {

«**AnadirPrestamos**» agrega un nuevo préstamo a objetc con el ID y NIF dado como parámetros. La fecha de la salida del préstamo se calcula con la fecha actual del sistema.

5. CONCLUSIONES

Mis conclusiones tras haber realizado este proyecto son:

Java debe mejorar la forma de crear interfaces. En mi opinión, Java no es tan fácil como. NET.

En. NET era imposible enlazar con el servicio web con "Servicio de Referencia". Tuve que usar "WebReference".

En Java, l han utilizado las listas matrices porque, en mi opinión, es más eficiente ya que ahorra consultas al servidor.

He hecho inserciones y consultas utilizando SQL y no las clases de persistencia.

En Java que he utilizado. Jsp (páginas de servicios de Java).

En. NET he utilizado una aplicación independiente.

www.sociedadelainformacion.com

Edita:

Director: José Ángel Ruiz Felipe
Jefe de publicaciones: Antero Soria Luján
D.L.: AB 293-2001
ISSN: 1578-326x