UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

ELABORACIÓN Y USO DE SERVICIOS WEB EN EL DESARROLLO DE HERRAMIENTAS PARA LA SISTEMATIZACIÓN DE LA INFORMACIÓN EN COLECCIONES BIOLÓGICAS.

REPORTE DE TRABAJO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

DANIEL JUÁREZ CRUZ

DIRECTOR DE TESIS:

DRA. AMPARO LÓPEZ GAONA

2013

1. Datos del alumno
   Juárez
   Cruz
   Daniel
   57 94 81 33
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Ciencias de la Computación
   404101280

2. Datos del tutor
   Dra
   Amparo
   López
   Gaona

3. Datos del sinodal 1
   Mat
   Salvador
   López
   Mendoza

4. Datos del sinodal 2
   M en I
   Gerardo
   Avilés
   Rosas

5. Datos del sinodal 3
   Dr
   José de Jesús
   Galaviz
   Casas

6. Datos del sinodal 4
   M en C
   Gustavo Arturo
   Márquez
   Flores

7. Datos del trabajo escrito
   Elaboración y usos de Servicios Web en el desarrollo de herramientas para la sistematización de la información en colecciones Biológicas.
   70 p
   2013

Índice

Introducción

México es uno de los países con mayor diversidad biológica en el mundo, principalmente en anfibios, reptiles, mamíferos y plantas. En 1929 fue abierto el Instituto de Biología de la UNAM y desde esa fecha se recolecta la información que forma parte de las Colecciones Biológicas.

La UNIBIO (Unidad de Informática para la Biodiversidad) del Instituto de Biología se encarga de la sistematización y manejo de esta información, de la creación de herramientas para el análisis y la consulta de los datos albergados en las colecciones, además de la publicación de estos en Internet para que se encuentren a disposición tanto del público como de los investigadores.

Otra de las actividades de la UNIBIO es la creación y administración de portales temáticos, compuestos por material producido por los investigadores del Instituto de Biología, con el objetivo de reflejen el estado actual de los mismos y de que se encuentren al día.

En general, todas las actividades de la UNIBIO están desarrolladas para la web y la información que alberga el instituto es bastante, ya que recolecta información desde 1929 y los datos están en constante crecimiento. Por estas razones se requiere de una tecnología que alimente de información a las herramientas y portales que se desarrollan de una forma rápida, correcta y dinámica.

Los Servicios Web son en gran medida, una tecnología que cubre los requerimientos, son una herramienta que se encuentra en la web y está disponible en cualquier momento para el que lo requiera. Una de sus ventajas es la interoperabilidad y lo fácil que resulta consumirlos. Son herramientas que están a nivel servidor y se utilizan principalmente para el flujo de información.

En la UNIBIO se trabaja con esta tecnología y gracias a ello, el manejo de los datos se realiza de una forma eficaz, rápida y altamente dinámica, ya que los portales requieren de mucha información y obtener estos desde el comienzo, puede llevar mucho tiempo y afectar directamente al usuario, además de que se corre el riesgo de una saturación y el colapso del sistema.

Las posibilidades de los Servicios Web son muchas y el propósito de este reporte es ejemplificar el uso de esta tecnología, analizar sus ventajas y desventajas, analizar algunos usos y explicar un poco el desarrollo de esta, todo enfocado al trabajo que se realiza en la UNIBIO y en mi experiencia como desarrollador web dentro de esta unidad.

En este trabajo se describen los Servicios Web: características, sus tipos, estructuras, funciones, ventajas, desventajas, etcétera. Una vez que se sabe lo que es un Servicio Web, se ejemplifican algunos usos y se analiza el consumo de esta tecnología dentro de procesos específicos.

Posteriormente se estudia el desarrollo de los Servicios Web en donde se describe el proceso y las acciones que realizan, se analizan las actividades y el proceso que se requiere para cumplirlas. Finalmente se habla de las posibilidades de convertir alguna herramienta ya desarrollada en un Servicio Web, se describe la herramienta y el proceso para la conversión a Servicio Web.

Para cubrir estos puntos se utilizan proyectos que realicé dentro de la UNIBIO que permiten evidenciar el papel que toman los Servicios Web dentro de cada proceso. Cada proyecto se analiza y documenta para ejemplificar algunas posibilidades de esta tecnología.

La estructura de trabajo está compuesta por los siguientes capítulos:

1. **Servicios Web**

Se analizan los Servicios Web a fondo, los requerimientos que puede cubrir, se estudian los tipos que existen y sus diferencias, las ventajas y desventajas con otras tecnologías, además de ejemplificar con situaciones comunes.

2. **Uso de Servicios Web**

Aquí se analizan algunos usos particulares de los Servicios Web, estos usos se abarcan utilizando proyectos que realicé dentro de la UNIBIO y que resuelven los requerimientos de forma satisfactoria. Los módulos que se describen en esta sección son los siguientes:

2.1. ***WSNewSpecies*.-** Es un módulo perteneciente al portal “Especies nuevas para la Ciencia descritas en el Instituto de Biología” (unibio.unam.mx/especiesnuevas/) el cual se encarga de alimentar al sitio de información además de la creación dinámica de las gráficas interactivas, los mapas y las fichas técnicas de las especies.

2.2. ***Taxonomía*.-** Este proyecto es un árbol taxonómico interactivo, el cual se alimenta de un Servicio Web para su constante crecimiento. Este árbol muestra los niveles taxonómicos y se va construyendo a partir de las peticiones del usuario, se alimenta de una base de datos con todos los registros del Instituto de Biología.

2.3. ***Diagnósticos*.-** Esta herramienta se encarga de obtener un análisis de las bases de datos, para esto utiliza gráficas, las cuales muestran el estado de los campos de una tabla en particular, esta herramienta tiene dos tipos de gráficas, una que muestra los campos y su porcentaje de nulos y otra en donde muestra una gráfica por cada campo en donde se grafica el valor más frecuente de ese campo.

3. Desarrollo de Servicios Web

Este capítulo tiene como objetivo ampliar la visón para la creación de Servicios Web, se describen los módulos de tal forma que se muestre de forma clara el proceso que realiza cada Servicio Web. Los proyectos que se utilizan son:

3.1. ***WSDataTools*.-** Se encarga de obtener tablas de frecuencia de una tabla en particular de alguna base de datos. Toma la tabla y crea una nueva tabla de dos columnas, una el registro y la segunda el número de frecuencias que tiene dentro de la tabla.

3.2. ***WSImageMosaic*.-** Es un módulo es un mosaico de imágenes que se construye con imágenes del acervo digital del Instituto de Biología y el cual, cada que se visita, muestra imágenes distintas ya que cada que se inicia, obtiene 7 fotografías de forma aleatoria, para esto utiliza un Servicio Web que le proporciona esas imágenes.

3.3. ***Queue*.-** Este proceso se encarga de encolar procesos los cuales son llamadas a Servicios Web y su función es evitar la saturación del servidor. Una vez que se encuentra lista la petición, notifica por correo electrónico al usuario.

4. Conversión de una herramienta a un Servicio web

Finalmente en este capítulo se analiza la posibilidad de convertir algún fragmento de software en un Servicio Web con todos los beneficios que eso conlleve. Los proyectos que ejemplifican este capítulo, son proyectos que nacieron de un módulo que no era un Servicio Web. Los proyectos son los siguientes:

4.1. ***PruebaFonetica*.-** Esta herramienta se encarga de analizar y decidir si dos palabras son fonéticamente iguales.

4.2. ***WSUnifica*.-** Esta es una herramienta que se encarga de la limpieza de bases de datos, principalmente en campos libres que normalmente contienen muchas palabras y no siempre están escritos de la misma forma. Esta herramienta se encarga de obtener posibles registros que representen lo mismo aunque no estén escritos de la misma forma, todo esto para unificar los registros de la base de datos.

La tecnología que se utiliza dentro de la UNIBIO, y por ende, en el desarrollo de estos proyectos, es la siguiente: Java, J2EE, JSP, HTML, CSS, PostgreSQL, PHPPgAdmin, Javascript, JSON, jQuery y NetBeans IDE. En algunos proyectos se utiliza tecnología adicional tales como bibliotecas o plugins que se mencionarán dentro del análisis según sea el caso.

1. Servicios Web

En la actualidad, los sistemas se caracterizan por ofrecer diversos servicios, los cuales pueden llegar a ser independientes y estar disponibles en el mismo lugar. Estos servicios pueden depender de procesos totalmente diferentes pero que al unirse los resultados, se obtiene una respuesta mucho más completa y funcional para el usuario, o incluso el conjunto de estos puede ser necesario para la ejecución de un proceso. La concentración de estos servicios puede llegar a aumentar la complejidad del sistema si consideramos una estructura independiente la cual realiza todos los procesos por sí misma y se ejecuta en un solo lugar.

Tomemos como ejemplo un portal dedicado a organizar eventos sociales (bodas, XV años, etcétera), el cual se encarga de organizar todo (iglesia, salón, banquete, música, etcétera). Si consideramos este portal con la estructura mencionada anteriormente, se debería tratar de una empresa la cual debe estar en constante comunicación con todos los servicios (las fechas disponibles de las iglesias en catálogo, así como los salones, los menús de cada chef con diversos tipos de comida y música variada), y para esto se deben de poner en contacto de alguna manera (por teléfono, mediante su página de internet, por fax, etcétera) por cada petición que realicen los usuarios. Ahora pensemos que estas peticiones son muy frecuentes y el portal siempre debe de tener la información actualizada, correcta y disponible, entonces cada que hay una petición se debe de solicitar la información, ya que si la petición anterior solicitó la misma información y reservó alguna fecha, esta debe de verse reflejada en las próximas peticiones para mantener la fiabilidad de la información. Este proceso junto con las formas de obtención de toda la información no es óptimo, no es fluido y está propenso a errores así como a la no disponibilidad de la información.

Ahora consideremos un portal el cual se encarga de buscar productos en diferentes tiendas y así obtener toda la información necesaria del producto en cuestión (saber en dónde está disponible, el precio en cada tienda, ofertas en caso de que existan en alguna tienda, etcétera), esto para realizar la mejor compra o incluso saber dónde se encuentra disponible el producto. Si nuevamente tomamos la estructura anterior en este sistema, se requiere primeramente una base de datos con los catálogos de cada tienda, además de estarlos actualizando constantemente dependiendo totalmente de todas las tiendas y su movimiento en los mismos (nuevos productos, ofertas, cantidad de estos en existencia, etcétera), todo esto para ofrecer resultados óptimos, vigentes y verdaderos al usuario, y justo ahí es donde encontramos lo complejo que podría resultar este sistema ya que si

consideramos por ejemplo el cambio de temporada en los productos o los días de ofertas en toda la tienda, el trabajo de mantener la base de datos actualizada o incluso con los datos correctos se torna en un trabajo muy laborioso y poco sustentable, incluso sin considerar el peor de los casos.

Con este par de escenarios utilizando una estructura independiente nos damos cuenta que esta no es la óptima para sistemas que realizan varias operaciones para obtener uno o varios resultados, que son constantemente dinámicos. Pensemos en una aplicación que se alimenta de otros lugares dentro de una red para obtener las peticiones que se le piden, cada uno de estos lugares, capaces de inter operar con cualquier otra aplicación y otorgar un servicio en específico a quien lo requiera, además de estar disponibles en cualquier momento. Estos lugares por llamarlos de alguna forma se les conocen como servicios web.

1.1. ¿Qué es un Servicio Web?

Un Servicio Web (Web Service) es una aplicación que se encuentra dentro de una red (principalmente internet) que con la ayuda de protocolos y estándares, intercambia información con otras aplicaciones de cualquier tipo. En un esquema donde se utilizan servicios web, existen dos tipos de aplicaciones: las que ofrecen el servicio que podría llamarse servidor, y las que solicitan el servicio que recibe el nombre de cliente. El proveedor o servidor es un procedimiento remoto, y el cliente solicita el servicio a través de la red.

Un recurso es cualquier cosa que se puede direccionar en la Web y los Servicios Web cumplen esta propiedad ya que se encuentran dentro de una red y tanto para acceder a ellos como para recibir la petición se utiliza principalmente el protocolo HTTP y se requiere de una dirección para la interacción cliente / servidor (véase la figura 1.1).

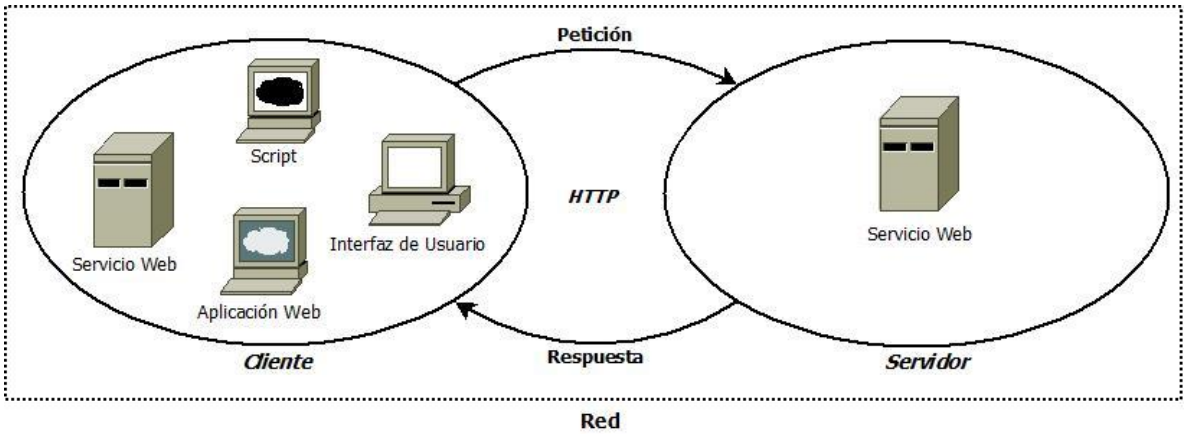

Figura 1.1: Esquema cliente – servidor con Servicios Web

En un sistema, un Servicio Web es una unidad independiente que se encarga de la implementación de uno o varios requerimientos, con esto podemos decir que un sistema que utiliza Servicios Web está compuesto de uno o varios recursos de los que depende el resultado final.

Uno de los principales objetivos de los Servicios Web es la interoperabilidad entre aplicaciones, esto sin importar en qué plataforma o lenguaje de programación se encuentren, qué aplicación cliente sea, ni mucho menos en donde se ejecuten. Todo esto para obtener resultados dinámicos, actualizados y correctos.

Otro de los principales objetivos es facilitar la expansión de cualquier aplicación que lo requiera, esto gracias a lo fácil que resulta acceder a un recurso y obtener los resultados sin preocuparnos por el proceso de obtención, lo único que debemos saber es cómo solicitar ese servicio (reglas, parámetros, etcétera). Si se modifica o expande alguno de ellos, esto no afectará la forma de acceder al recurso, esto solo será alterado cuando se modifique la posición del sistema dentro del servidor que lo contenga.

La estructura de un sistema que utiliza Servicios Web puede llegar a ser muy compleja, es un sistema modular en donde cada modulo puede ser un conjunto de instrucciones en donde se ejecuten diversos recursos y estos a su vez dependan de otros, un ejemplo genérico se muestra en la figura 1.2.

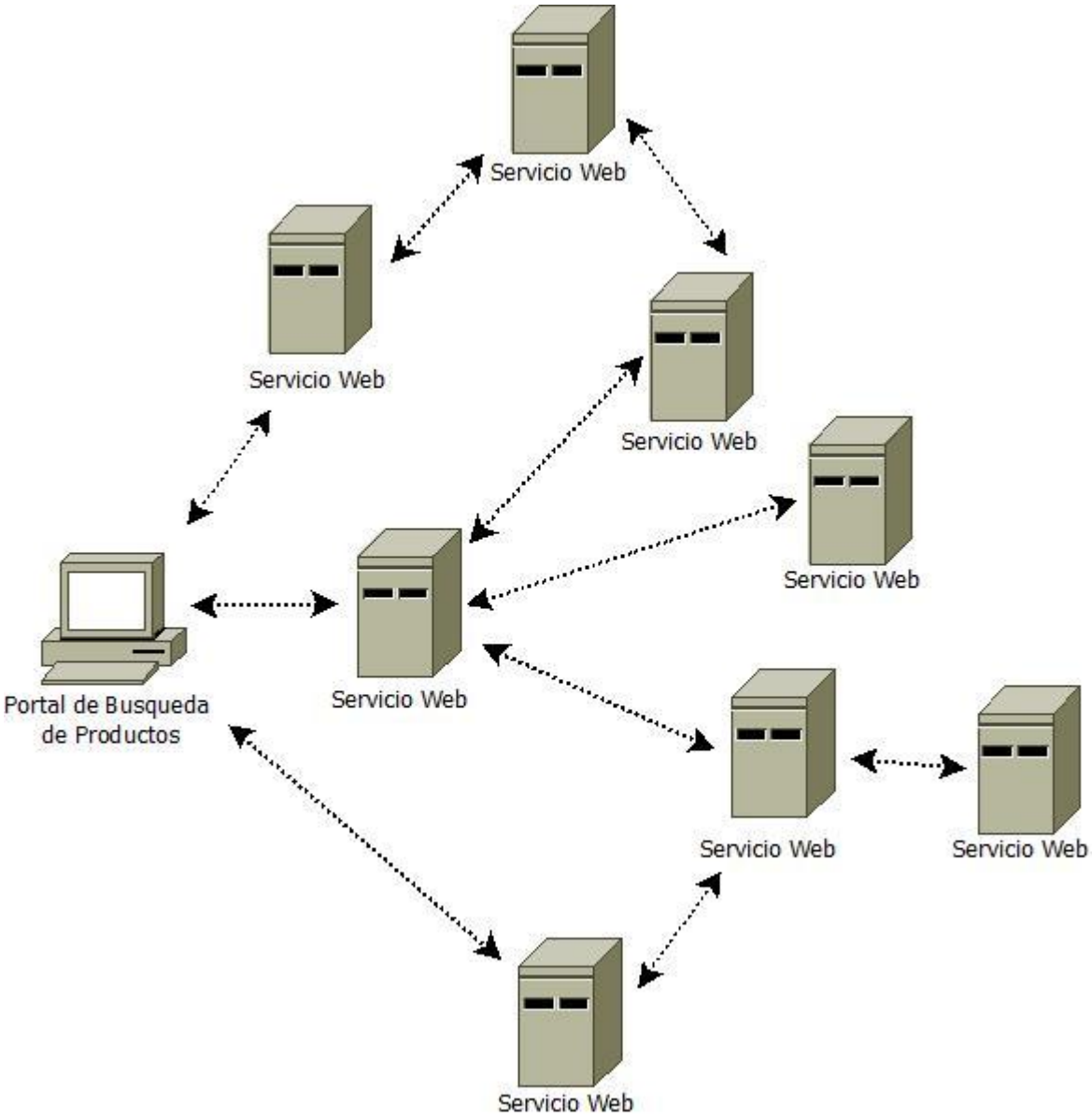

Figura 1.2: Ejemplo de un sistema modular.

Para el proceso de ejecución de un sistema así, debe de realizarse con un determinado orden ya que algunas tareas pueden depender de la ejecución de un recurso y este a la vez de algunos otros. Inclusive varias tareas pueden depender de un mismo recurso. En general, el resultado que se obtiene de la ejecución del sistema depende de un conjunto de recursos que a su vez dependen de otros, creando una red de comunicación entre ellos.

Retomemos y analicemos los escenarios anteriores con el uso de Servicios Web:

Pensemos en el portal que organiza eventos sociales como cliente de servicios web, en donde cada elemento requerido para organizar un evento es un servicio web. Con esta visión, cada que el usuario haga una petición, el portal obtendrá los datos de esta a partir de los servicios web de cada rubro. Al hacerlo de esta forma, la obtención de los resultados será muy cómoda, fácil y dinámica, además de tener la certeza de que lo que se muestre como resultado a la petición estará actualizado y correcto. Solo nos preocuparemos por el manejo de la información que nos arrojen los servicios web de cada elemento del evento.

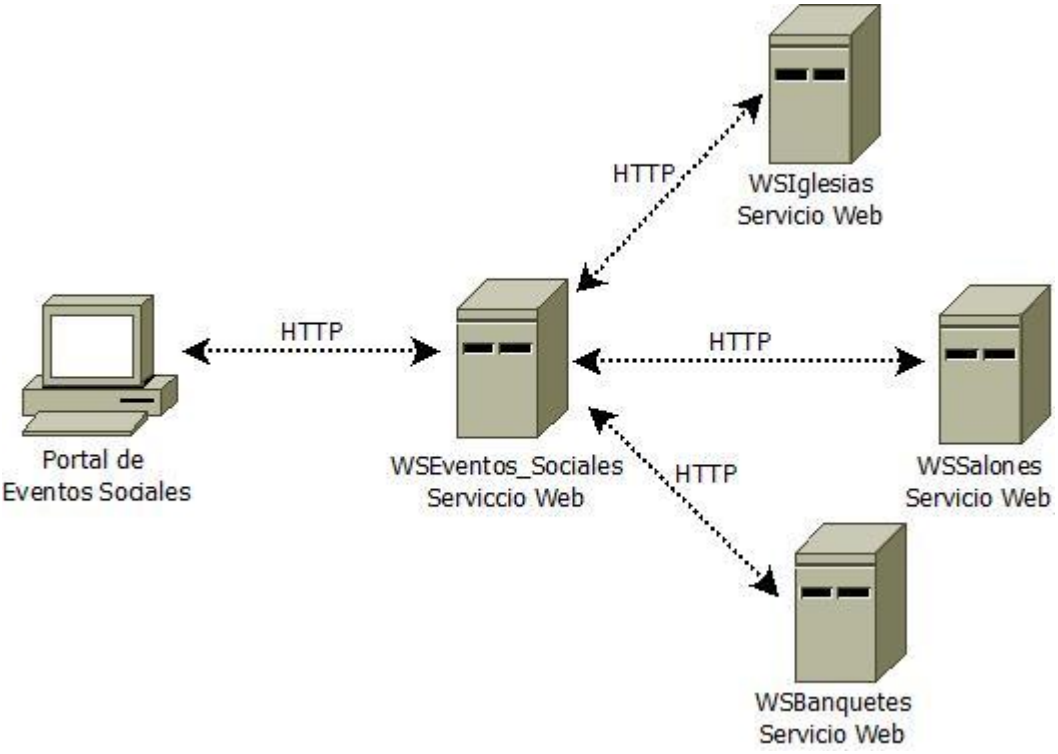

Figura 1.3: Diagrama del ejemplo del portal de Eventos Sociales

En la figura 1.3 observamos que el proceso inicia desde el portal que es donde se realiza la petición, por lo tanto, toma el papel de cliente. El servidor sería WSEventos_Sociales ya que recibe la petición y la procesa para regresar la respuesta. Pero a su vez, este requiere de otros recursos para crear el resultado, así que manda la petición correspondiente a los demás recursos para finalmente formar la respuesta de la petición principal, por lo tanto WSEventos_Sociales también toma el papel del cliente dentro del sistema. También los otros recursos pueden requerir de otros mas y convertirse en clientes además de servidores, pero eso no lo sabremos y no lo necesitamos.

Por último retomemos el ejemplo del portal de búsqueda de productos y pensemos en que cada tienda que se encuentra en este portal tiene su propia base de datos que se va actualizando conforme cada tienda venda sus productos, se surtan de nueva mercancía y/o apliquen ofertas en diversos productos, además de un Servicio Web que proporciona toda la información de su catálogo. Así el portal ya no requiere tener una base inmensa en donde se encuentra centralizada toda la información de cada tienda. Ahora, por cada petición que los usuarios hagan, basta con que el portal solicite a cada tienda la información a partir de sus servicios web, utilizando como parámetro la consulta del usuario. Finalmente obtendremos la información de cada tienda sin la necesidad de tener localmente la información, y con la certeza de que los datos que se obtienen son correctos.

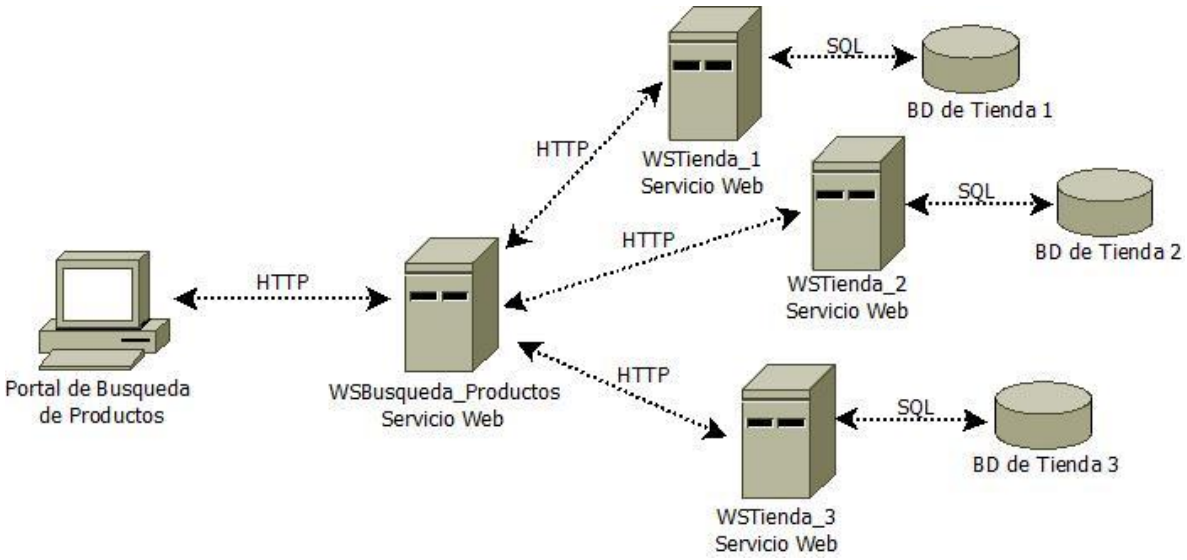

Figura 1.4: Diagrama del ejemplo del portal de búsqueda de productos

En la figura 1.4 la distribución de los datos, el recurso WSBusqueda_Productos únicamente centraliza los datos para mostrarlos al cliente pero este no tiene localmente los datos y tampoco los almacena, únicamente los manipula para presentar la respuesta a la petición. Los recursos externos de cada tienda son los que consultan la base de datos para obtener los datos requeridos en cada petición. Finalmente, cada que el cliente realice una petición, el sistema llamará a cada tienda, a su vez cada una de ellas realizará las consultas necesarias para la obtención de los datos y esto nos garantiza que los datos estarán actualizados.

Con este par de ejemplos, nos damos cuenta que los Servicios Web atacan principalmente el dinamismo de la información y son excelentes herramientas para lograrlo pero además de eso, aportan una gran cantidad de ventajas dentro de un sistema.

1.2. Características

Hoy en día existen principalmente dos tipos de Servicios Web: los que están basados en SOAP (Simple Object Access Protocol) y los que están basados en REST (Representational State Transfer). El primero es un protocolo y el segundo es un estilo de arquitectura de software que está fundado en el protocolo HTTP, pero analicemos cada uno de ellos respectivamente.

1.2.1. SOAP

Es un protocolo estándar, que se encarga de comunicar a diferentes procesos por medio del intercambio de mensajes encapsulados en archivos XML. SOAP surgió a partir del protocolo creado por David Winer en 1998, llamado XML-RPC, ya que empresas importantes tales como IBM Y Microsoft se interesaron por la propuesta de Winer y decidieron desarrollar un protocolo basado en este.

El objetivo era crear un protocolo para la invocación de servicios remotos, el cual estuviera basado en protocolos estándares de internet (HTTP para la conexión y medio de transporte, XML para la codificación de datos) con una independencia en la plataforma de desarrollo.

Un Servicio Web basado en SOAP utiliza un WSDL, (Web Services Description Language) que es un formato XML en donde se describe todo lo relacionado al servicio (la forma de comunicación, las funciones disponibles, el formato de los mensajes para las peticiones así como los tipos de datos que recibe y el tipo de salida). Este documento es independiente al desarrollo y plataforma del servicio que funciona como documentación para su uso y descripción.

Figura 1.5: Mensaje SOAP

La comunicación se realiza mediante mensajes XML con un formato especifico en donde se almacena la petición y la respuesta, se compone de un sobre que representa la raíz del mensaje, dentro, se encuentra el encabezado que contiene la información acerca de la codificación de los tipos de datos, forma de procesamiento y contenido de identificación. El cuerpo que también se encuentra dentro del sobre, y contiene todos los datos codificados y la información para la ejecución y respuesta de la petición (figura 1.5).

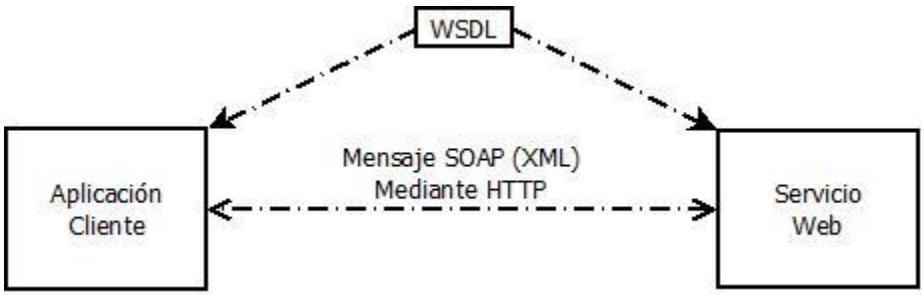

Figura 1.6: Diagrama SOAP

En la figura 1.6 se muestra la estructura y función de SOAP. Dentro del esquema tenemos al cliente y al servidor que en este caso es un Servicio Web, se observa que es un ciclo que comienza con la petición del cliente la cual está codificada en un mensaje XML que viaja por medio del protocolo HTTP hasta llegar al servidor. Posteriormente, el servidor devuelve la respuesta con las mismas condiciones con las que fue enviada la petición y el cliente procesa la respuesta. Finalmente se observa que el archivo WSDL está

presente en los dos extremos, del lado del cliente como manual para la conexión al servicio y del lado del servidor para reflejar cambios internos y retroalimentación.

1.2.2. REST

Es un estilo de arquitectura que se basa en HTTP, utiliza las funciones de este protocolo (GET, POST, PUT y DELETE) para la interacción cliente-servidor, y cada recurso es representado por una URI distinta. Esta idea surgió en el año 2000 por Roy Fielding el cual introdujo este término en su tesis doctoral y es uno de los principales autores de la especificación de HTTP.

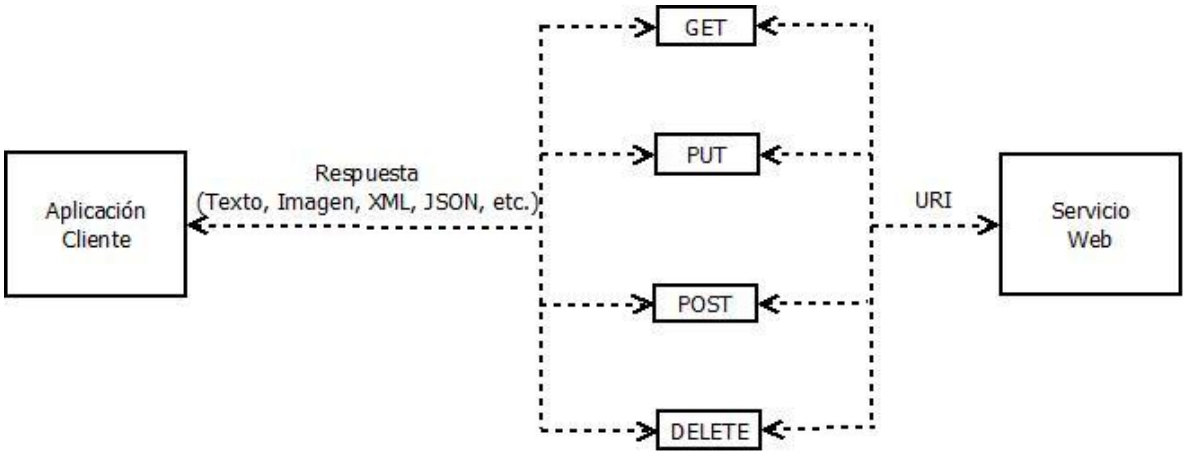

Figura 1.7: Diagrama REST de un Servicio Web

Analizando la estructura y proceso que muestra la figura 1.7, Servicio Web basado en REST funciona a partir de URI´s (Uniform Resource Identifier) las cuales son únicas para cada recurso (servicio) y contienen toda la información para conectarse a este. Estas URI's tienen la forma de una URL pero dentro de una URI agregamos los parámetros necesarios, entre otras cosas, para la ejecución del recurso. Esta URI es enviada a través de las funciones HTTP (GET, POST, PUT y DELETE) y el recurso obtiene de esta, toda la información necesaria para ejecutarse. Una vez que el Servicio Web procesa la petición, devuelve el resultado de la petición igualmente mediante HTTP, esta respuesta no es de un tipo en especifico, puede variar ya que no se requiere de un formato especial. Finalmente cada recurso tiene un pequeño manual en donde se indica al usuario como invocar el recurso (este puede ser una pagina jsp, html, etcétera).

1.2.3. Diferencias entre los tipos de Servicios Web

Los dos tipos de Servicios Web utilizan estructuras distintas y el proceso difiere entre ellos, algunas diferencias entre ellos son:

- En REST, las operaciones se definen en los parámetros de la URI mientras que en SOAP, las operaciones son definidas como puertos WSDL.
- REST es una estructura que no requiere la especificación de tipos, mientras que SOAP requiere el tipo de cada parámetro.
- REST maneja una URI por cada instancia del proceso mientras que SOAP maneja una sola dirección para todas las instancias.
- Ambas usan HTTP, la diferencia es que SOAP solo lo utiliza como medio de transporte, mientras que REST adopta toda la ideología del protocolo.
- SOAP trabaja exclusivamente con XML, la petición y la respuesta son XML mientras que en REST, los recursos pueden tener más representaciones además de XML, tales como HTML, texto, JSON, etcétera. Esto nos muestra lo flexible que es REST a la hora de la interoperabilidad.

Estos son los principales tipos de Servicios Web. A partir de este momento, solo haremos referencia a los servicios web basados en REST, por su fácil construcción, acceso y rapidez a la hora de la interoperabilidad, además de la flexibilidad para mostrar los recursos y manipularlos.

1.3. Ventajas

Ahora analicemos las ventajas de los Servicios Web:

- **Accesibilidad**

Como ya lo hemos mencionado, los Servicios Web, se encuentran disponibles dentro de una red, lo cual implica que siempre están disponibles y acceder a ellos es muy sencillo, lo único que debemos saber es la dirección de servicio y los parámetros que requiere para su funcionamiento.

- **Compatibilidad**

Los servicios Web no necesitan cumplir ningún requerimiento técnico para realizar la interoperabilidad, esto es: el cliente puede ser una aplicación hecha en un lenguaje totalmente ajeno al lenguaje en el que está hecho el Servicio Web que este requiere, y la comunicación entre ellos se realizará sin problemas y sin ningún tipo de consideración

extra. Esto se logra ya que al cliente no le importa en dónde se ejecuta el recurso, cómo obtiene los datos y mucho menos cómo está hecho, a él solo le importa el resultado. De la misma forma, al Servicio Web no le importa qué tipo de aplicación lo llame, lo que haga con los datos, y tampoco la forma en que los consume, solo verifica que la llamada y los parámetros sean correctos.

- **Resultados dinámicos**

La respuesta que da un recurso, depende de la fuente de la que se alimenta, y la variación de la respuesta depende totalmente de que tanto se modifique la fuente (bases, de datos, documentos, listas de datos, etcétera). Cada que se modifiquen los datos de donde se alimenta el recurso, al realizar la petición, este mostrará inmediatamente los cambios realizados. Esto puede aportar dinamismo a un portal que muestra los datos, ya que cada que se consulte el portal, estará actualizado y con los cambios que se realicen desde la fuente. Todo esto se realiza desde el servidor, el cliente no se preocupa por esto.

- **Manipulación de datos**

Los Servicios Web tienen la capacidad de recibir los parámetros en diferentes estructuras, desde arreglos; hasta formatos más elaborados como un XML o un JSON, de los cuales hablaremos más adelante, y a su vez, la respuesta también puede tomar cualquier estructura, esto depende del tipo de datos y queda a criterio del desarrollador. También depende de la función que utilizaremos para el envío de datos, GET utiliza directamente la URI para el envío de parámetros y el POST se utiliza más dentro de un formulario. Con esta característica, el cliente y el servidor pueden enviar o recibir una gran cantidad de datos y el consumo de estos será muy fácil y rápido.

- **Facilidad de implementación**

Una de las diferencias que tiene una aplicación común con un Servicio Web es justamente que el segundo, se encuentra disponible en una red, y esto hace que otros clientes que se encuentren dentro de la red puedan consumir el recurso. Convertir una aplicación en un Servicio Web es sencillo, se requiere definir las funciones HTTP disponibles para el recurso, establecer la URI única para este recurso, algunos Annotations del protocolo y finalmente un pequeño manual para describir la forma de interactuar con el recurso, normalmente es un documento html o jsp.

1.4. Desventajas

Para hablar de las desventajas de los Servicios Web, hay que hablar sobre los sistemas distribuidos que son un grupo de hardware y software distribuidos en una red que se comunican y coordinan para realizar un objetivo y que se presentan como un solo sistema. Algunas de sus características principales son:

- **Concurrencia:** Se pueden acceder a las partes del sistema de forma simultánea.
- **Transparencia:** Se trata de mostrar un único sistema sin evidenciar los fragmentos de este, el usuario no sabrá que está utilizando un sistema distribuido.
- **Escalabilidad:** Un sistema distribuido es capaz de crecer sin aumentar su complejidad ni reducir su rendimiento.
- **Tolerancia a fallos:** Tiene la capacidad de recuperarse cuando falle alguno de sus componentes y asegurar el funcionamiento continuo.
- **Persistencia:** Todos los elementos comparten un estado y llevan una continuidad en la ejecución del proceso.

Después de dar una pequeña introducción de lo que son los sistemas distribuidos y de analizar todo lo que se ha explicado en este capítulo, podemos decir que un sistema que utiliza Servicios Web cumple con la definición de un sistema distribuido, además de tener algunas de las características mencionadas de estos sistemas, y es en este punto donde resaltan las desventajas de usar Servicios Web. Consideremos un sistema que usa Servicios Web como un sistema distribuido; y las desventajas que tiene son:

- No cuentan con persistencia, funcionan a partir de peticiones y el tiempo de vida es el tiempo que tarden en ejecutarse, no tienen noción del estado del sistema en el que son solicitados y son hechos exclusivamente para una tarea en particular. Debido a que usan HTTP como medio de comunicación, no se mantiene un estado de conexión.
- Cuando llega a fallar un Servicio Web, el cliente (en este caso un programa) que depende de este, puede llegar a manejar excepciones pero esto no cubre la ausencia del recurso y el resultado final se ve afectado, por lo tanto, no existe la tolerancia a fallos.
- Estos sistemas introducen mayor sobrecarga en la red ya que operan con peticiones en modo texto y otros sistemas distribuidos, operan en modo binario, esta hace más pesada la transacción y en la concurrencia puede llegar a afectar al servidor en donde se encuentre el recurso.

En conclusión, un sistema que utiliza Servicios Web es un sistema distribuido, el cual carece de persistencia dentro de los recursos pero el cliente es el encargado de esta tarea. Destacan su interoperabilidad, ya que cualquier Servicio Web puede interactuar con cualquier otro Servicio Web independientemente del lenguaje en el que se hayan desarrollado. Y finalmente, el hecho de que estos utilicen el protocolo HTTP, hace que funcionen correctamente dentro de internet ya que toda la infraestructura disponible, está diseñada principalmente para el tráfico de este protocolo, incluyendo las configuraciones de seguridad (cortafuegos).

2. Uso de Servicios Web

Como hemos analizado en el capítulo anterior, el principal objetivo que tiene un Servicio Web es la interoperabilidad y al usarlos obtenemos grandes ventajas tales como: proporcionar el contenido de algún sitio, dotar alguna herramienta con información dinámica, obtener una gran cantidad de información en una sola consulta, etcetera. Para todas estas tareas, además de un Servicio Web, se requiere de una estructura de datos robusta que permita el almacenamiento de grandes cantidades de datos, un fácil acceso a ellos y versatilidad en los tipos de datos, esa estructura es JSON.

JSON (JavaScript Object Notation) es un formato ligero, ideal para el intercambio de datos, es fácil de leer y escribir y puede ser interpretado en cualquier lenguaje. Está compuesto por pares etiqueta/valor y su estructura es la siguiente:

```
JSON = {
        etiqueta1: valor1,
        etiqueta2: valor2,
        …
        etiquetaN: valorN
}
```

Los tipos de datos de los valores pueden ser de cualquier tipo, desde un entero, hasta otro JSON, esto es, se pueden almacenar valores de diferentes tipos en un mismo JSON tal como el siguiente ejemplo:

```
JSON = {
        numero: 275,
        cadena: “Hola Mundo!”,
        arreglo: [5,4,3,2],
        json: {et1: 3, et2: 1, et3: 2}
}
```

En general para guardar información en un JSON, basta con almacenar el valor dentro de una etiqueta “JSON(etiqueta) = valor” y para obtener el contenido basta con solicitarlo con la respectiva etiqueta “valor = JSON.etiqueta”.

En este capítulo, nos enfocaremos en el uso de los Servicios Web y cómo es que se beneficia el sistema utilizándolos. Analizaremos algunos proyectos en los que realice herramientas para el consumo de estos y en los cuales se muestran algunos usos de los Servicios Web, los proyectos son:

- WSNewSpecies
- Taxonomía
- Diagnósticos

2.1. WSNewSpecies

“WSNewSpecies” es un módulo que forma parte del portal de consulta llamado “Especies nuevas para la Ciencia descritas en el Instituto de Biología” (figura 2.1) el cual muestra, a partir de gráficas estadísticas, mapas y listas, la densidad de las especies nuevas descritas en el Instituto de Biología, esto se presenta por país, estado (sólo México), género, entre otras características. Es un portal que se actualiza de forma autónoma, cada que se editan o registran nuevos datos, se reflejan en el portal debido a que se crea en tiempo real con el estado actual de la base de datos.

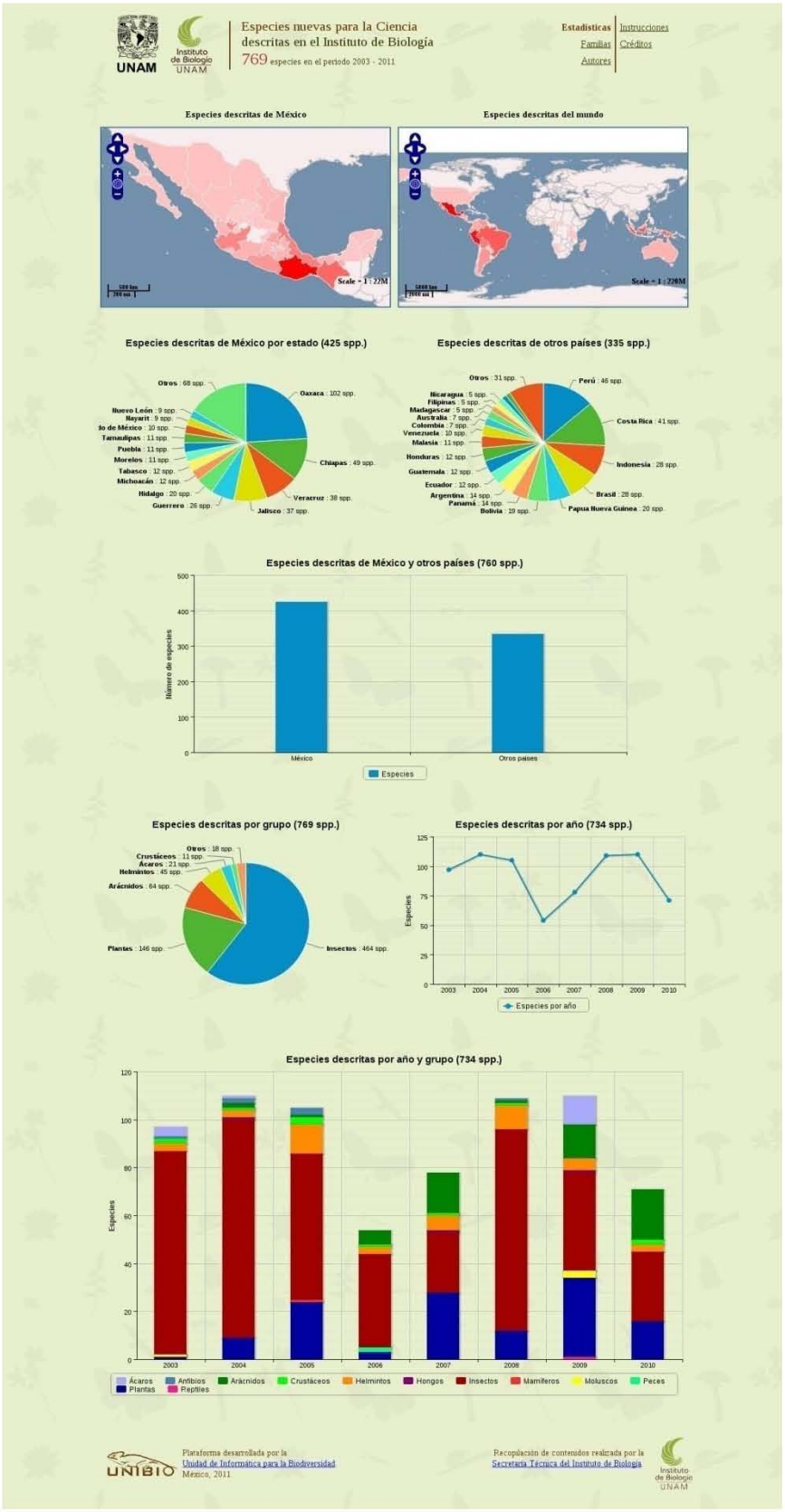

Figura 2.1: Portal “Especies nuevas para la ciencia descritas en el Instituto de Biología”

WSNewSpecies está compuesto por tres recursos (Servicios Web): “WSSpeciesData”, “WSSpeciesCollection” y “WSExecuteQuery”, un script “charts” y un archivo de propiedades “config_cfg”. Finalmente utiliza el plugin “Highcharts” que está hecho en JQuery.

2.1.1. Objetivo

Este módulo se encarga de alimentar al sitio de toda la información que se muestra en él, es el responsable de recolectar los datos de la base de datos y además de dar funcionalidad a todos los elementos del portal. Obtiene la información para mostrar la densidad de especies nuevas dentro de mapas, así como también el listado de familias, géneros, especies e información completa de fichas de cada especie. En general, los datos para el llenado del sitio.

Otra función de este módulo es la implementación de gráficas dinámicas hechas con el plugin “Highcharts”, las cuales se encargan de mostrar estadísticas de especies nuevas además de filtrar y consultar los datos de forma más sencilla para el usuario.

2.1.2. Funcionamiento

El portal “Especies nuevas para la Ciencia descritas en el Instituto de Biología” está compuesto por 4 módulos, cada uno de ellos utiliza algún Servicio Web del módulo “WSNewSpecies” para su construcción.

El primer módulo está compuesto por mapas que se muestran al inicio de la página. Estos mapas representan la cantidad de especies nuevas que tiene cada región, ya sea por país o por estado de la República Mexicana. Estas regiones se representan con una tonalidad rojiza, esta tonalidad depende de la densidad de especies nuevas que se encuentren en esa región, entre más intenso sea el rojo, es mayor el número de especies nuevas registradas en esa región, véase la figura 2.2.

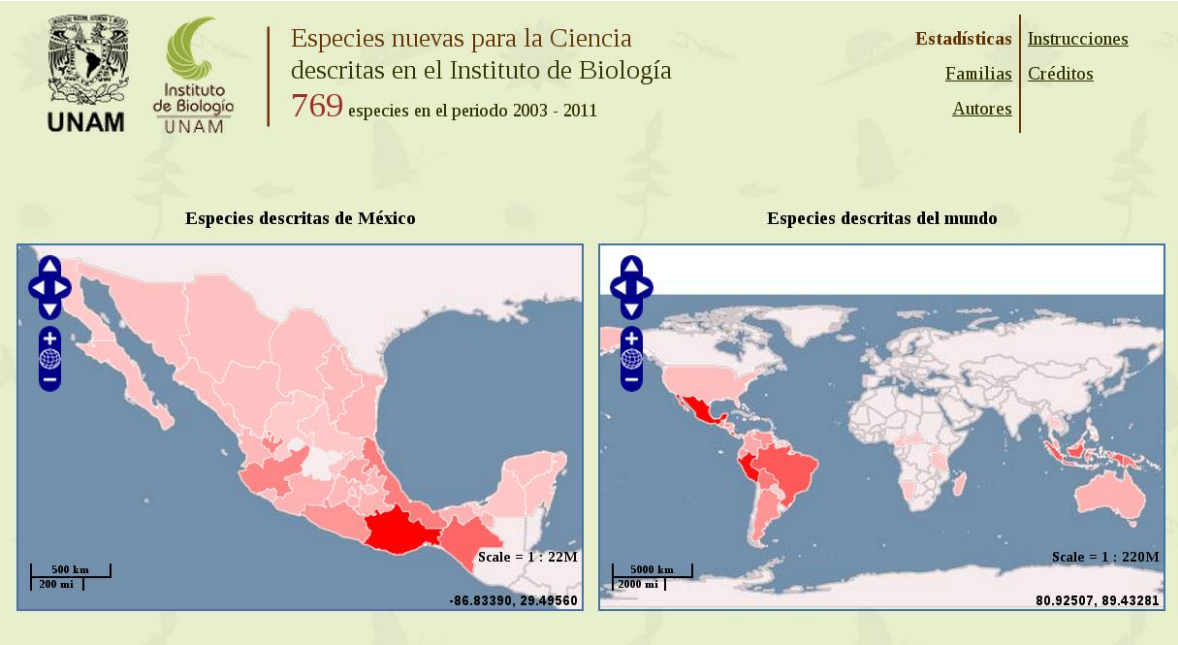

Figura 2.2: Mapas que muestran la densidad de las especies nuevas.

Para determinar el color de cada región, se realizan dos peticiones al recurso “WSSpeciesData”, una para cada mapa. La forma en la que se invoca el recurso es muy sencilla, sólo requiere de un parámetro, un JSON que contiene la especificación del tipo de búsqueda, por países o por estados (únicamente estados de México). Una vez que se realiza la petición, el recurso crea la conexión a la base de datos y obtiene la lista de los estados o países junto con la cantidad de especies nuevas descritas en él. Esta lista es ordenada de forma descendente y es enviada como respuesta para el cliente.

Una vez que se obtiene la lista de los territorios junto con su cantidad correspondiente de especies nuevas, se determina a partir de las cantidades, la tonalidad de cada región. Este proceso se realiza cada vez que se carga el portal, por lo tanto, cada que se registren o eliminen nuevas especies, los mapas muestran los cambios de forma automática así como todo lo que se crea a partir de algún recurso dentro del portal, el proceso se muestra en la figura 2.3.

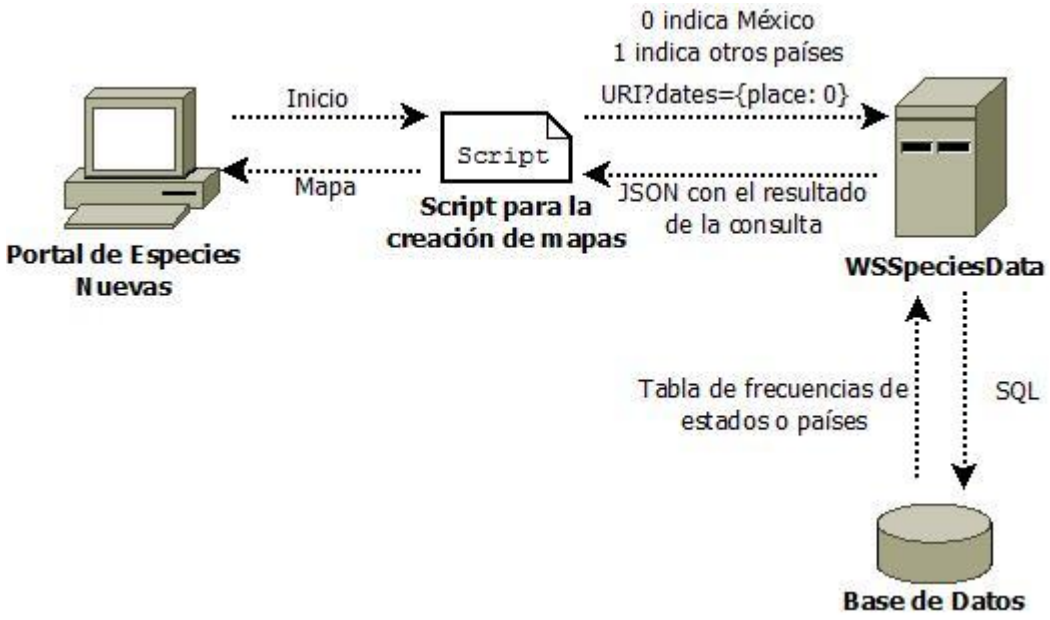

Figura 2.3: Proceso de la obtención del mapa de los estados de México dentro del portal de especies nuevas.

La segunda parte por analizar son las gráficas expuestas en el portal. Representan filtros para la consulta de especies nuevas, muestran estadísticas y son interactivas. Estas gráficas se crean utilizando el plugin de jQuery “Highcharts” y el recurso ”WSExecuteQuery”, todo esto dentro del script de Javascript “charts”.

Dentro del portal se exponen distintos tipos de gráficas, algunas se señalan en la figura 2.4. Cada una se construye de diferente manera ya que los requerimientos son distintos y su estructura cambia. Debido a esto, por cada gráfica, existe una función dentro de “charts”, en donde se realiza todo el proceso para su elaboración.

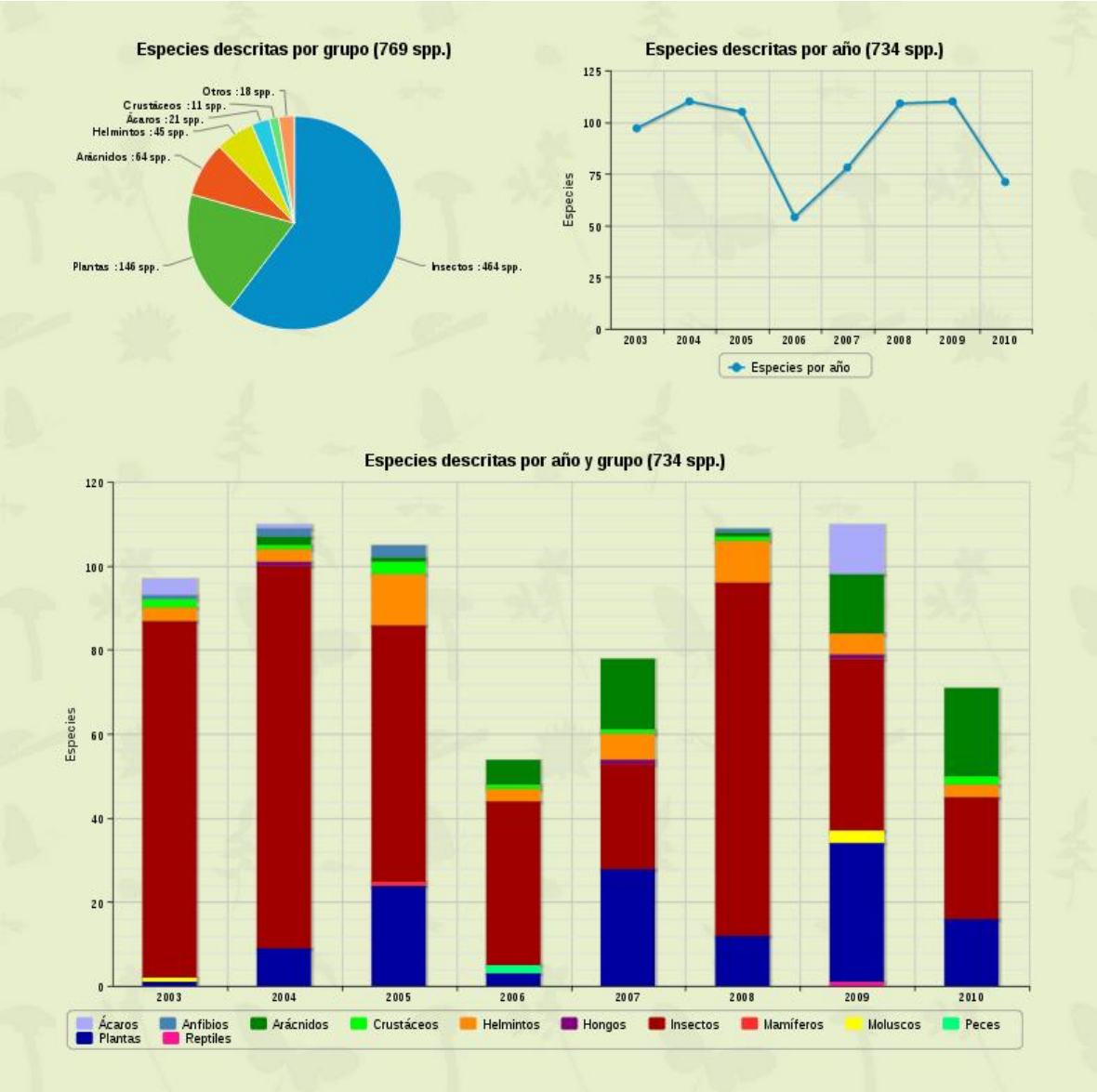

Figura 2.4: Gráficas estadísticas para las especies nuevas.

La creación de las gráficas inicia con la recolección de los datos de cada una, para eso requieren de al menos una consulta SQL. El recurso “WSExecuteQuery” se encarga de ejecutar esas consultas y como respuesta devuelve una tabla que representa el resultado. Este recurso recibe como parámetro una cadena que contiene la consulta SQL y otra cadena que indica el nombre de la base de datos en donde se ejecuta la consulta.

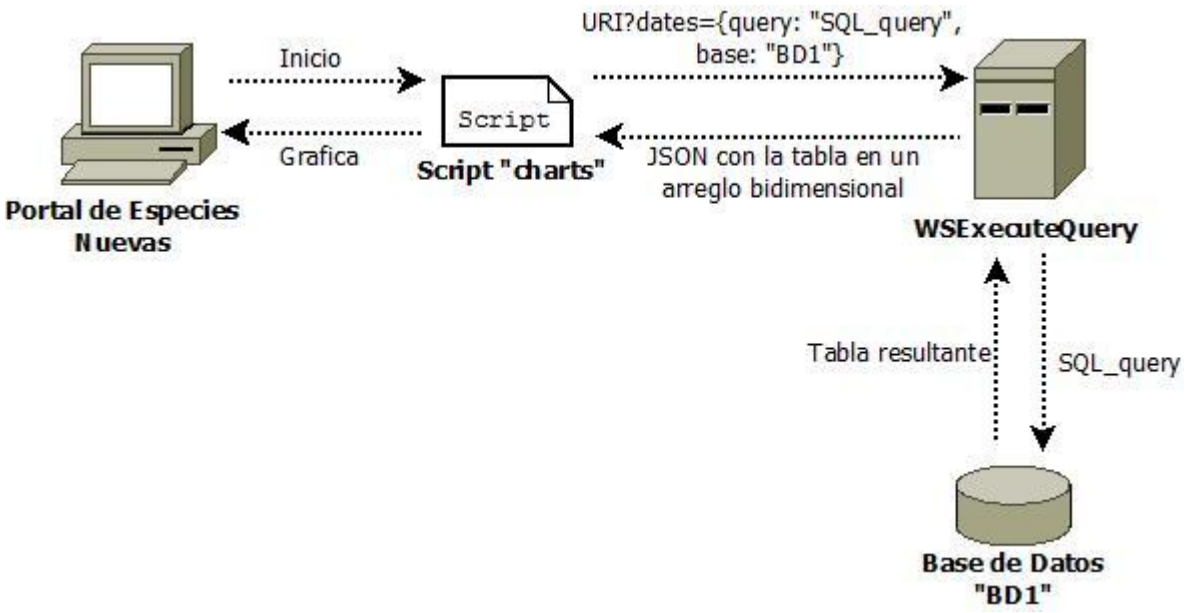

Figura 2.5: Proceso para la obtención de una gráfica en el portal de especies nuevas.

Una vez que se tienen los datos que muestran las gráficas, como se observa en la figura 2.5, el script “charts” manipula y almacena los datos dentro de una estructura correcta para ejecutar el plugin “Highcharts” (esta estructura de datos depende del tipo de gráfica). En algunos casos, es necesario realizar operaciones con los datos para obtener la información que presentan las gráficas, un ejemplo de esto, son las gráficas de tipo circular, ya que estas muestran porcentajes y estos son calculados a partir de los datos obtenidos.

Finalmente se crea la gráfica a partir del plugin “Highcharts” y las estructuras creadas anteriormente. Este plugin debe de configurarse dependiendo del tipo de gráfica, para eso se utiliza una estructura de tipo JSON en donde se especifican todas las características de la gráfica incluyendo la información a graficar. Este proceso se realiza para cada una de las gráficas.

Como se mencionó anteriormente, estas gráficas son interactivas ya que el usuario puede activar o desactivar la información mostrada y seleccionar alguna área de éstas para obtener listas de ejemplares. Funcionan como filtros de información, cuando se elige algún elemento de alguna gráfica, se despliega la lista de los ejemplares que cumplen con las características que nos arroja gráficamente, por ejemplo: si seleccionamos la rebanada que pertenece a Brasil dentro de la gráfica “Especies descritas de otros países”, este nos desplegará una lista con todos los ejemplares que fueron descritos en Brasil, ahora si seleccionamos a los Helmintos en la barra del año 2007 dentro de la gráfica “Especies

descritas por año y grupo", nos devolverá la lista de arácnidos descritos en el año 2005 (figura 2.6).

Figura 2.6: Lista de ejemplares del grupo Helmintos, descritos en el año 2005.

Estas listas son creadas a partir de consultas SQL, las cuales son ejecutadas utilizando nuevamente el recurso "WSExecuteQuery". Cada que se presiona algún elemento de una gráfica, se crea la consulta correspondiente y se realiza la petición para obtener la lista de los ejemplares correspondientes (figura 2.7).

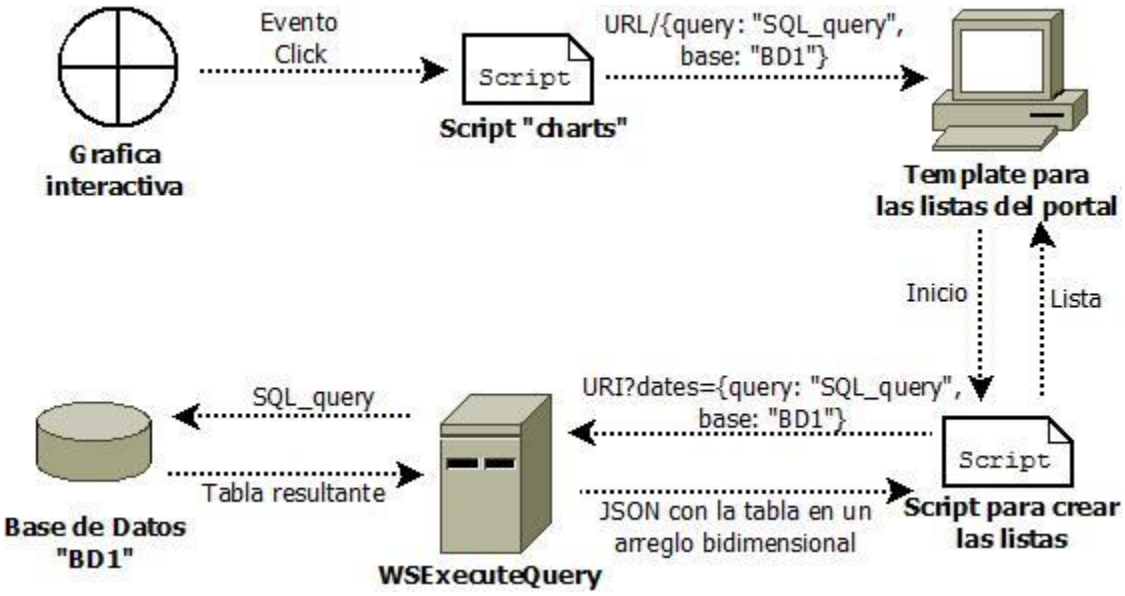

Figura 2.7: Proceso para la obtención de la lista de ejemplares a partir de las gráficas.

La tercera parte por revisar abarca las secciones “Familias” y “Autores” dentro del portal, al igual que ocurre cuando seleccionamos una parte de alguna gráfica, estas secciones también muestran una lista con familias y autores respectivamente, las gráficas muestran listas de ejemplares.

La diferencia de las listas que se obtienen de las gráficas a las listas de estas dos secciones, es que las secciones son listas que representan árboles en donde las hojas son las especies. Cada sección representa un árbol distinto, en el caso de “Familias”, el árbol es de tres niveles; familias, géneros y especies. En el caso de “Autores”, el árbol cuenta únicamente con dos niveles; autores y especies. Para recorrer estos árboles, el usuario selecciona un elemento de la lista, al seleccionarlo, se obtiene la lista del siguiente nivel, y así hasta llegar a las hojas del árbol que son los ejemplares.

Para la obtención de estas listas se utiliza el recurso “WSSpeciesCollection”, los parámetros que recibe pueden variar, y dependiendo de lo que recibe, es el método que se ejecuta. La razón de esto es que la consulta que realiza a la base de datos, no siempre es la misma, depende de lo que se necesite, puede devolver listas de familias, géneros, especies y la información de una especie, además de una lista con los autores descriptores.

En el caso de la sección "Familias", cuando el usuario da clic en esa parte, se realiza la petición al recurso, esta petición se ejecuta sin algún parámetro y con esto, nos devuelve la lista de todas las familias; cuando se selecciona una de la lista, se vuelve a

realizar una petición pero ahora lleva como parámetro la selección del usuario, esto nos devuelve una lista con los géneros de esa familia. Al final, al seleccionar un género, se realiza la petición con la familia y el género como parámetros y obtenemos los ejemplares de ese género y familia. Lo mismo ocurre con la sección "Autores", solo que al inicio se envía un indicador para identificar la solicitud de autores (figura 2.8).

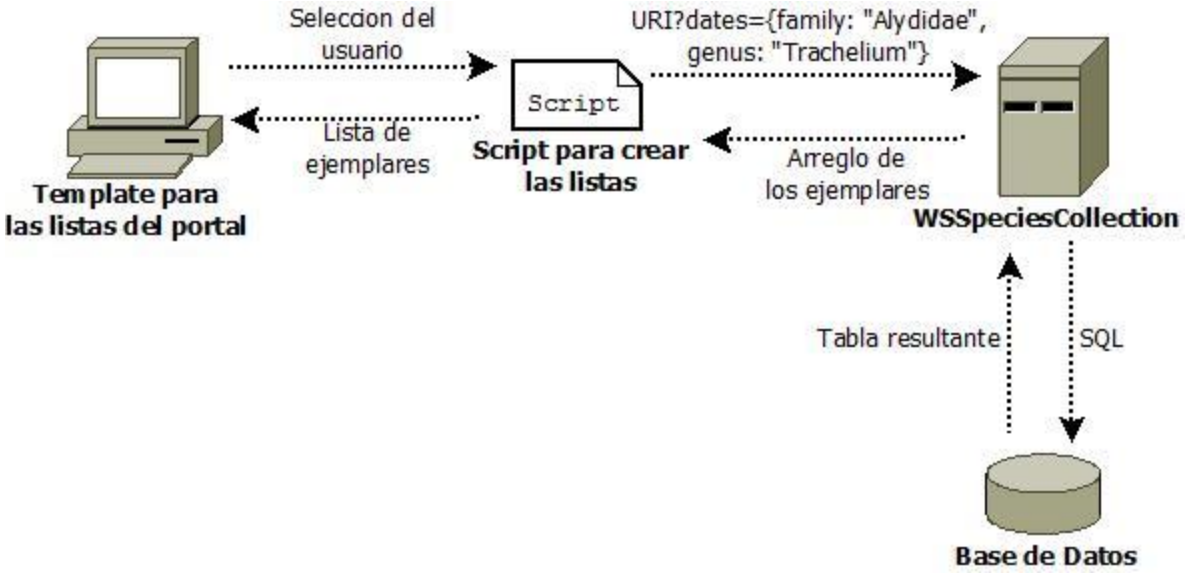

Figura 2.8: Obtención de la lista de las especies con familia “Alydidae” y genero “Trachelium”

Una vez que se obtiene la lista de ejemplares por cualquier medio disponible ("Familias", "Autores" o gráficas), el usuario tiene disponible la ficha técnica de todas las especies obtenidas, y esa es la cuarta y última sección del portal. Esta sección utiliza nuevamente el recurso "WSSpeciesCollection".

Cuando el usuario selecciona un ejemplar de la lista se realiza una petición al recurso y este arroja toda la información del ejemplar, se envía como parámetro el nombre del ejemplar y nos devuelve toda la información dentro de un JSON. Una vez que se obtiene la información, se crea la ficha técnica en donde el usuario puede consultar la información o incluso descargarla en un archivo PDF (figura 2.9).

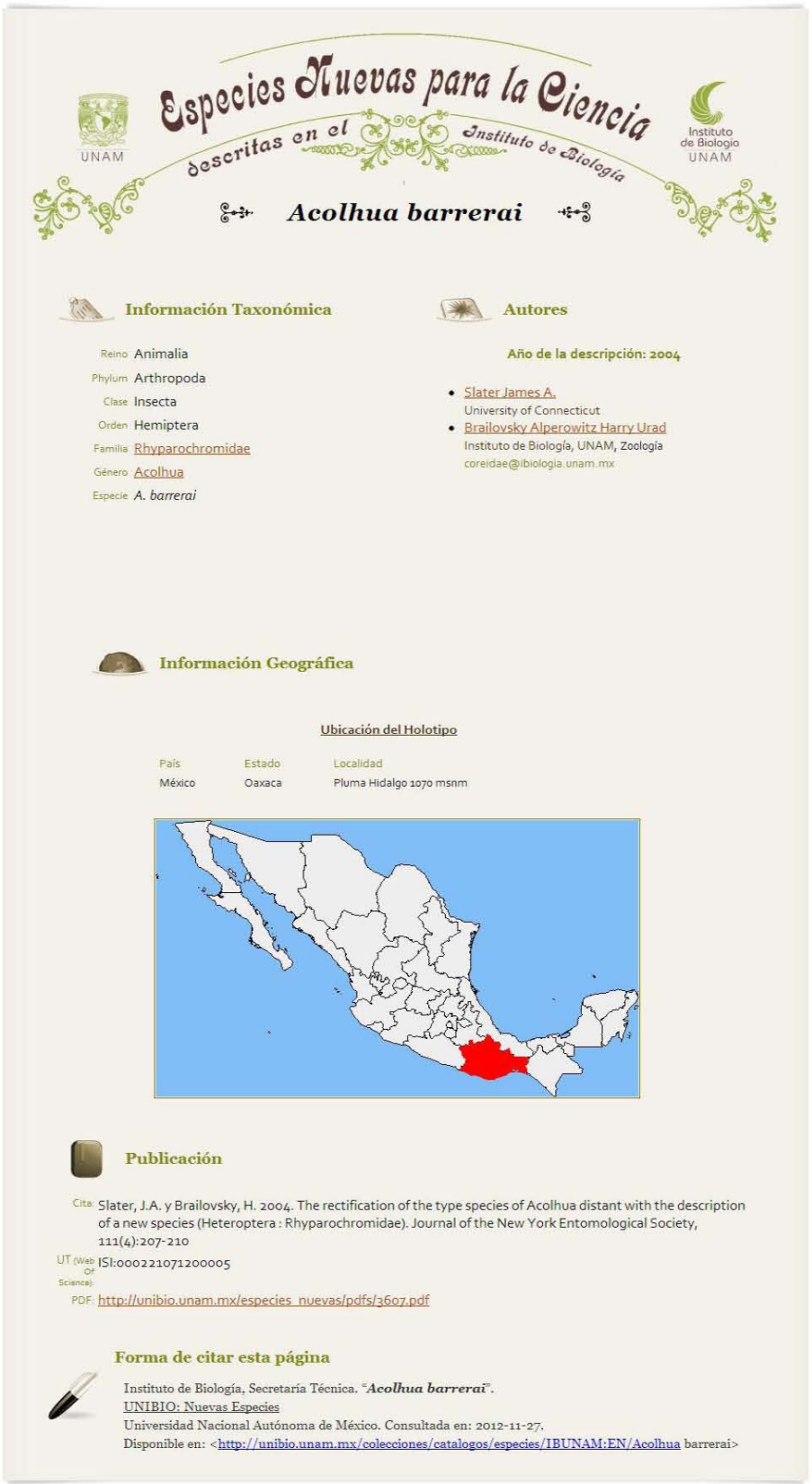

Figura 2.9: Ficha técnica del ejemplar "Acolhua Barrerai"

2.1.3. Ventajas

El portal de Especies Nuevas es totalmente autónomo, desde los mapas, pasando por el listado de las familias, géneros, especies, etcétera, así como las gráficas que siempre muestran las estadísticas actualizadas, dependiendo únicamente de la base de datos, si ésta se modifica, el sitio reflejará esos cambios de forma autónoma e inmediata. Otra ventaja es la forma de interacción que ofrece al usuario, ya que las gráficas son muy claras, explícitas y además sirven para consultar las especies.

2.1.4. Desventajas

El portal está compuesto en su totalidad por la información que se encuentra almacenada en la base de datos y que nos proporcionan los recursos que componen este módulo. Todos ellos realizan consultas, por un lado "WSExecuteQuery" requiere de una consulta como parámetro, la cual se crea del lado del cliente, este recurso se creó principalmente para la obtención de recursos de las gráficas. Por otro lado se tienen los recursos “WSSpeciesCollection” y “WSSpeciesData” que se encargan del resto de elementos, a diferencia del recurso anterior, estos no dependen de la consulta SQL ya que el encargado de crearla y ejecutarla son los recursos. Algo malo que muestran estos dos recursos son la redundancia, ya que se puede crear un recurso general de ambos y así generalizar uno de ellos y reducir el tamaño del módulo además de mejorar y robustecer uno de ellos, obteniendo al final un recurso dinámico que atiende diferentes tareas y requerimientos.

2.1.5. Problemas

Cada gráfica que se expone en el portal, requiere de una consulta, para obtener los datos necesarios para mostrarla, además de un número de datos diferentes ya que las gráficas son distintas.

Al principio, se optó por realizar un recurso por cada consulta ya que todas las consultas eran diversas, pero eso aparte de ser engorroso y repetitivo, afectaba el rendimiento y el tamaño del módulo. Las similitudes que tendrían todos esos recursos eran bastantes dado que todos realizaban una consulta y regresaban el resultado que siempre sería una tabla. Fue ahí donde se decidió crear un nuevo recurso, el cual se encargaría de ejecutar cualquier consulta y regresaría la tabla sin importar las características de esta, y así es como se realizó el recurso “WSExecuteQuery”. Otra ventaja de esto es que ese recurso es bastante genérico y fácil de usar, por lo que puede ser útil en cualquier otro proyecto.

2.2. Taxonomía

La taxonomía es el orden que se le da a las especies dentro de un sistema de clasificación y donde se manejan jerarquías. Los niveles taxonómicos que se utilizan en el Instituto de Biología son:

1. Reino
2. Filo
3. Clase
4. Orden
5. Familia
6. Genero
7. Especie

El módulo “Taxonomía” muestra un árbol interactivo con la taxonomía de todas las especies descritas en el Instituto de Biología. El árbol empieza mostrando los reinos y va creciendo conforme se va interactuando con él, pasando por todos los niveles taxonómicos hasta llegar a las especies. Tiene la capacidad de guardar ramas para disminuir el tamaño y así dejar abiertas sólo las ramas de interés. Contiene también tiene una versión no interactiva, la cual es en forma circular y está abierto hasta el nivel taxonómico “orden”, mostrando el número de especies que hay en cada rama.

Está compuesto por dos templates html: “interactivo” e “index”, cada uno contiene un script el cual crea el árbol correspondiente (“interactivo” corresponde a la gráfica interactiva e ”index” se encarga del árbol circular). Utilizan la biblioteca “Protovis” de Javascript que está hecha para la visualización además de utilizar el recurso “ReturnNivelTaxon”.

2.2.1. Objetivo

Utilizando la amplia base de datos con la que cuenta el Instituto de Biología, se crea un árbol interactivo con el objetivo de informar y mostrar los niveles taxonómicos de los ejemplares que se encuentran registrados en la base de datos. La interacción juega un papel muy importante, ya que tanto el usuario que sabe qué busca como el que simplemente lo usa sin algún objetivo definido de inicio, puede tener una gran utilidad dado que la información se encuentra perfectamente ordenada y cumple la función como herramienta de consulta.

2.2.2. Funcionamiento

El modulo contiene dos estructuras creadas con Javascript y la biblioteca “Protovis”, ambas son árboles pero la presentación varía además de que uno es fijo y el otro comienza desde la raíz, en este último, el usuario se encarga de ir abriendo los nodos según su interés. La construcción de éstas es muy similar porque las dos estructuras son árboles y utilizan los mismos datos, la diferencia radica en la obtención de los datos que son proporcionados por el recurso “NivelTaxon” debido a que lo realizan de distinta forma, el árbol que es fijo obtiene los datos en una sola petición mientras que el interactivo realiza tantas como el usuario requiera, por cada clic que reciba un nodo por primera vez, es una petición al recurso. Por esta razón analizaremos únicamente la construcción del árbol interactivo (figura 2.10) y posteriormente la obtención de los datos del árbol fijo.

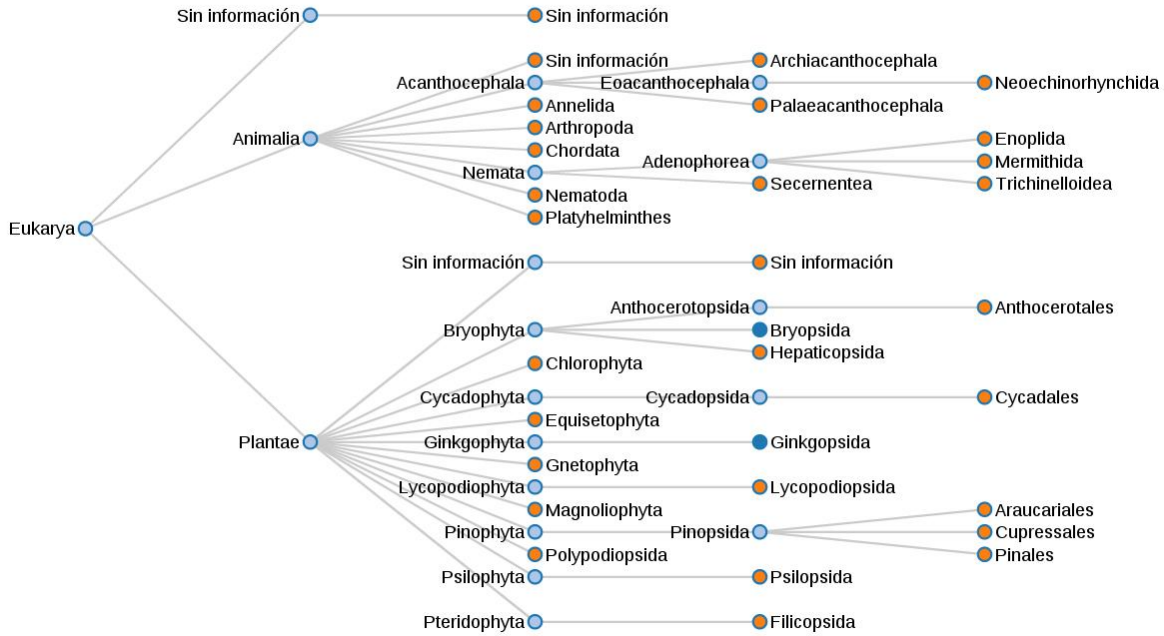

Figura 2.10: Árbol taxonómico interactivo.

Antes de describir el proceso para la creación del árbol interactivo, es importante analizar el recurso “NivelTaxon”. Este recurso requiere de un arreglo de cadenas como parámetro, nos devuelve un JSON muy particular que se detallará más adelante. El arreglo que recibe como parámetro debe de ser de una longitud mayor o igual a cero y de tamaño máximo a seis, este arreglo representa una ruta dentro del árbol y el rango se debe al

número de niveles del árbol, que es el mismo al número de niveles dentro de la taxonomía.

Cuando se realiza la petición con un arreglo de longitud cero, se obtienen los nodos hijo del nodo raíz (Eukarya), en otras palabras, se obtienen los nodos del primer nivel de la taxonómica, en este caso “Reino”. El arreglo tiene una estructura muy específica y sus posibilidades son las siguientes:

0 [] “Reino”

1 [“Reino”] “Filo”

2 [“Reino”,”Filo”] “Clase”

3 [“Reino”,”Filo”,”Clase”] “Orden”

4 [“Reino”,”Filo”,”Clase”,“Orden”] “Familia”

5 [“Reino”,”Filo”,”Clase”,“Orden”,“Familia”] “Genero”

6 [“Reino”,”Filo”,”Clase”,“Orden”,“Familia”,“Genero”] “Especie”

Como pasa en los árboles, no se puede obtener los nodos de un nivel sin antes pasar por un nivel superior, esto es, no podemos obtener los hijos de “Orden” sin saber a que “Filo” pertenece y mucho menos a que “Reino”.

En el caso de la respuesta del recurso, nos regresa un JSON con los hijos del nivel correspondiente, pero su estructura nuevamente es particular, ya que los nodos son devueltos en forma de etiqueta dentro de un JSON, y no como valor de alguna etiqueta, como valor nos devuelve cero. A continuación la estructura:

```
{
    “hijo 1”: 0,
    “hijo 2”: 0,
    …
    “hijo n”: 0
}
```

El cero es una bandera que se utilizará en la construcción del árbol. Ahora, se va a analizar la construcción del árbol.

Como se mencionó anteriormente, estos árboles están creados con la biblioteca “Protovis” la cual está basada en Javascript y la forma en la que se pasan los parámetros es a partir de un JSON con la siguiente estructura:

```
{“nodo 1”:
        {“hijo 1.1”:0,
        “hijo 1.2”:
                {“hijo 1.2.1”: 0,
                “hijo 1.2.2”: 0},
        “hijo 1.3”: 0}
“nodo 2”: 0,
“nodo 3”: 0}
```

En este ejemplo se muestra que “nodo 1”, “nodo 2” y “nodo 3” se encuentran al mismo nivel y son hermanos, “nodo 1” tiene tres nodos hijos, “nodo 2” y “nodo 3” no cuentan con nodos hijos. Dentro de los hijos de “nodo 1” observamos que “hijo 1.2” también tiene hijos: “hijo 1.2.1” e “hijo 1.2.2. Y con esta estructura es como le mandamos los datos a “Protovis” para que genere el árbol con esa información. Los nombres de los nodos pueden tener como valor un número u otro JSON. Cuando tienen un valor, entonces son hijos o están ocultos, cuando tienen un JSON entonces están abiertos y tienen hijos. Los nodos que son padres tienen un tono azul claro y los nodos que no lo son toman un tono naranja.

Cuando se inicia el módulo comienza la raíz (Eukarya) y el primer nivel (Reino), para esto se realizó la petición al recurso con un arreglo vacío como parámetro y fue el que tomó “Protovis” para crear el árbol. Este es un JSON que contiene como etiquetas a los elementos del nivel Reino y como valores cero, a partir de esta estructura construye el árbol. Cuando el usuario abre un nodo nuevo (un nodo nuevo es aquel que no ha recibido ningún clic), se obtiene la ruta de ese nodo y se almacena en un arreglo que es con el que se realiza la petición. Cuando se obtiene la respuesta, sustituye el valor cero por el JSON de respuesta y se recarga el inicializador de “Protovis”. Este proceso se realiza siempre que el nodo sea nuevo (figura 2.11).

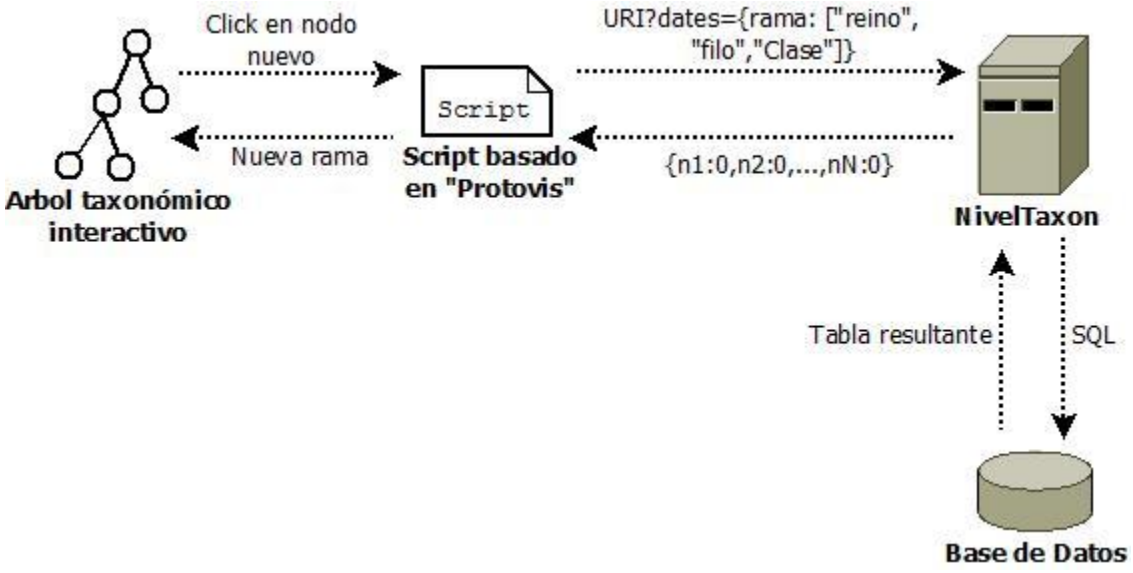

Figura 2.11: Proceso para la obtención de nuevos nodos dentro del árbol taxonómico interactivo.

Esta gráfica tiene la posibilidad de ocultar ramas que ya han sido abiertas, y así como fueron ocultas, volver a abrirlas en el estado en el que estaban. Basta con darle clic a un nodo padre, este tomará un tono azul rey indicando al usuario que ya ha visitado ese nodo.

Para realizar esa función se utiliza un cache, el cual es una copia de la estructura que va creando el usuario. Cuando el usuario esconde una rama, el nodo toma el valor uno y así se distingue de los nodos nuevos ya que estos tienen el valor de 0 y con esto se evita realizar otra petición al recurso y preservar el estado de la rama. Posteriormente se reinicia “Protovis” para mostrar los cambios.

Cuando el usuario da clic a un nodo ya visitado, el sistema busca ese nodo dentro del cache y sustituye el valor uno por el JSON que tiene el mismo nodo dentro del cache. Cada que se realiza una petición al recurso “NivelTaxon”, es porque se le dio clic a un nodo nuevo y las adiciones de ramas se realizan en ambas estructuras, pero cuando se esconde una rama, esos cambios no se realizan en el cache, únicamente se realizan en la estructura que utiliza “Protovis”. Los nodos visitados toman un color azul rey como se muestran en la figura 2.10 con el nodo “Bryopsida” o “Ginkgopsida”.

Al final, cuando se visita un nodo del nivel “Especie”, se envía la información para mostrar las referencias del ejemplar.

En el caso del árbol fijo (figura 2.12), únicamente se realiza una petición al recurso “NivelTaxon” con un número que representa el nivel del árbol que se creará el árbol. Al

final la respuesta es la que utiliza “Protovis” para crear la gráfica. Los nodos del último nivel muestran el total de especies que pertenecen a esa rama.

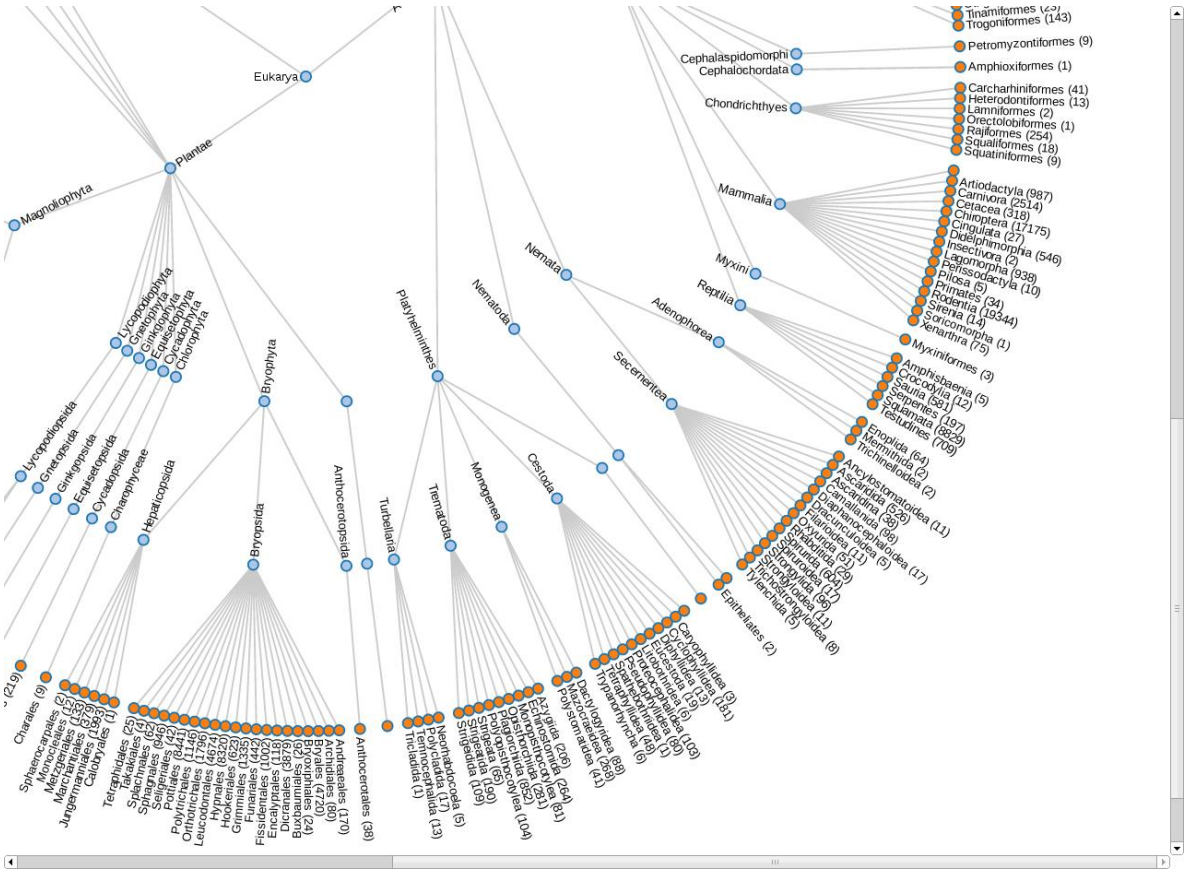

Figura 2.12: Árbol taxonómico radial con 4 niveles.

2.2.3. Ventajas

La base de datos de la que se alimenta este módulo contiene 565,977 especies con su información taxonómica, si se piensa en cargar toda la información desde el principio, afectaría directamente la carga inicial del módulo de forma considerable, además de cargar información excesiva y que no sería usada. Con el uso del recurso “ReturnNivelTaxon” todo eso se soluciona, ya que se carga únicamente la información que el usuario requiere y la fluidez de la herramienta es excelente y muy ligera.

2.2.4. Desventajas

Una desventaja quizá sea el número de conexiones a la base de datos, ya que por cada clic a un nodo, se realiza una conexión a la base, esto puede afectar bastante al sistema si no se tienen las medidas correspondientes y una administración robusta de conexiones.

Esta herramienta opera con una base de datos muy grande ya que se encuentra gran parte de los ejemplares descritos con los que cuenta el “Instituto de Biología”, por esta razón se consideró la función de poder ocultar ramas ya que el árbol puede crecer exponencialmente y la experiencia de usuario se puede ver afectada. Debido a esto, la estructura que representa el árbol puede crecer bastante y además de eso se tiene el cache que es mayor o igual de grande que la estructura principal. Esto puede causar problemas de memoria y se incrementa conforme el usuario visita nodos nuevos. Es un problema que tal vez cause la caída del servicio, pero las ventajas que da, hace que valga la pena correr el riesgo.

2.2.5. Problemas

Las posibilidades de “Protovis” son muchas ya que es una biblioteca amplia y robusta, se pueden realizar cosas muy llamativas e interactivas y su API está basada principalmente en ejemplos y así es como se aprende a utilizar esta biblioteca. Desafortunadamente no se cuenta con un ejemplo en el que la gráfica se fuese armando en el momento, había gráficas que tenían toda la información de inicio y solo escondían las ramas. Finalmente haciendo uso de “JQuery” y funciones extra de “Protovis” se pudo lograr la construcción de la gráfica en tiempo real.

Otro de los problemas fue la persistencia, ya que el árbol requería guardar estados, debido a la posibilidad de esconder ramas, en un inicio eliminaba lo que se encontraba en la rama y cuando el usuario quería volver a abrir esa rama, realizaba una nueva consulta, lo cual reiniciaba la rama y si el usuario tenía más de un nivel abierto, no se mostraba y no había forma de saberlo. Después se penso en el uso de otra estructura que hiciera la función de cache pero se necesitaba un método para saber cuáles son las ramas ocultas ya que puede haber sub ramas ocultas de una rama oculta y ahí fue donde se optó en utilizar la bandera del cero y el uno como valor de los nodos y así diferenciar los tipos de nodos.

2.3. Diagnósticos

El modulo “Diagnósticos” realiza gráficas estadísticas de una tabla perteneciente a una base de datos en particular, por ahora realiza gráficas para saber el porcentaje de nulos que tiene cada columna y también los registros más frecuentes de las mismas.

Está compuesto por dos recursos: “ConnectDB” y “SetData”, una plantilla HTML “template_diagnosticos”, un archivo de propiedades y un script “constructCharts”, además de utilizar la biblioteca “Highcharts” de Javascript.

2.3.1. Objetivo

La función de este módulo es mostrar el estado de los datos dentro de las tablas de una base de datos, para esto utiliza gráficas para realizar estadísticas de los datos y reflejar anomalías o errores en los registros, redundancia de la información, cambios, actualizaciones, entre otros.

Las gráficas que utiliza son de barras y cuenta con dos tipos: gráfica de no nulos la cual grafica el porcentaje de nulos con los que cuenta cada columna de la tabla, el 100% indica que la columna no cuenta con valores nulos (figura 2.13). La otra gráfica es de frecuencias, esta tabla muestra los diez registros con mayor frecuencia dentro de una columna, se encuentran ordenados de mayor a menor y se muestra el número de repeticiones (figura 2.14).

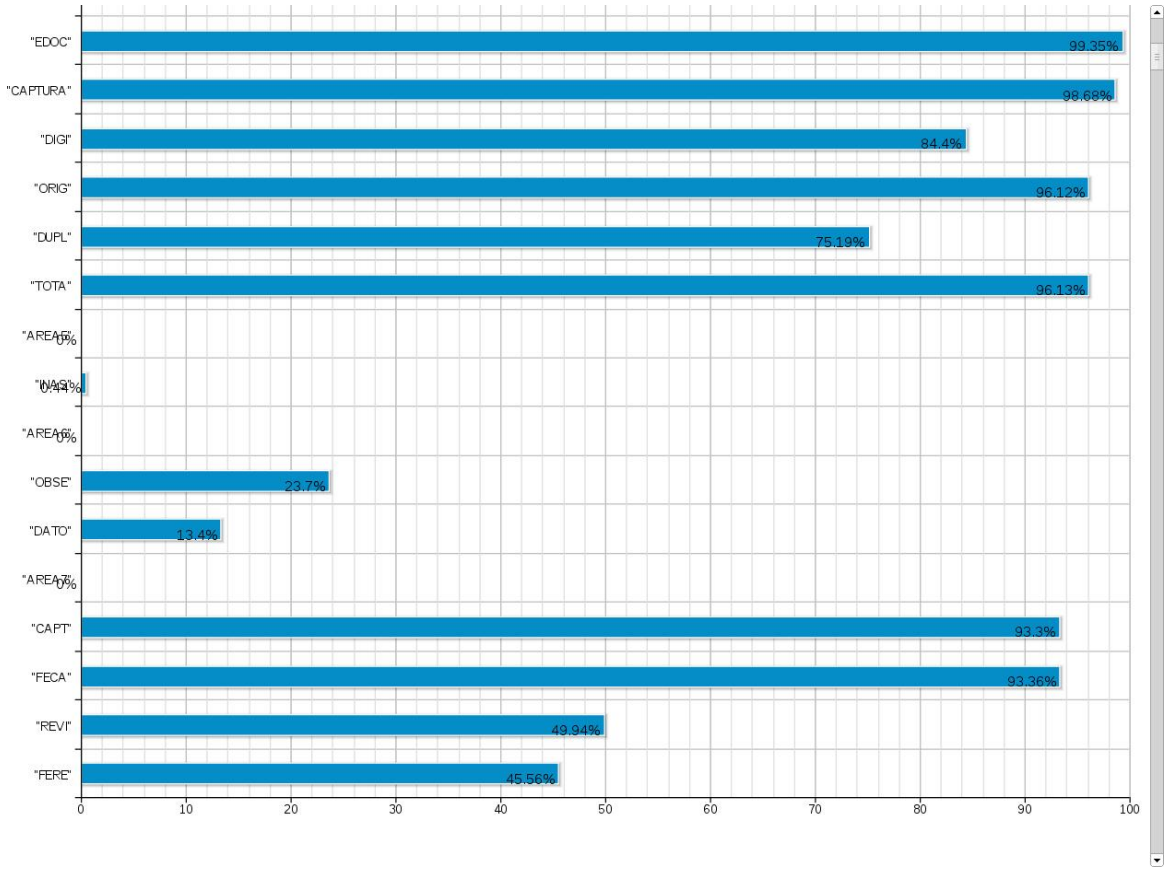

Figura 2.13: Gráfica de no nulos para el modulo “Diagnósticos”

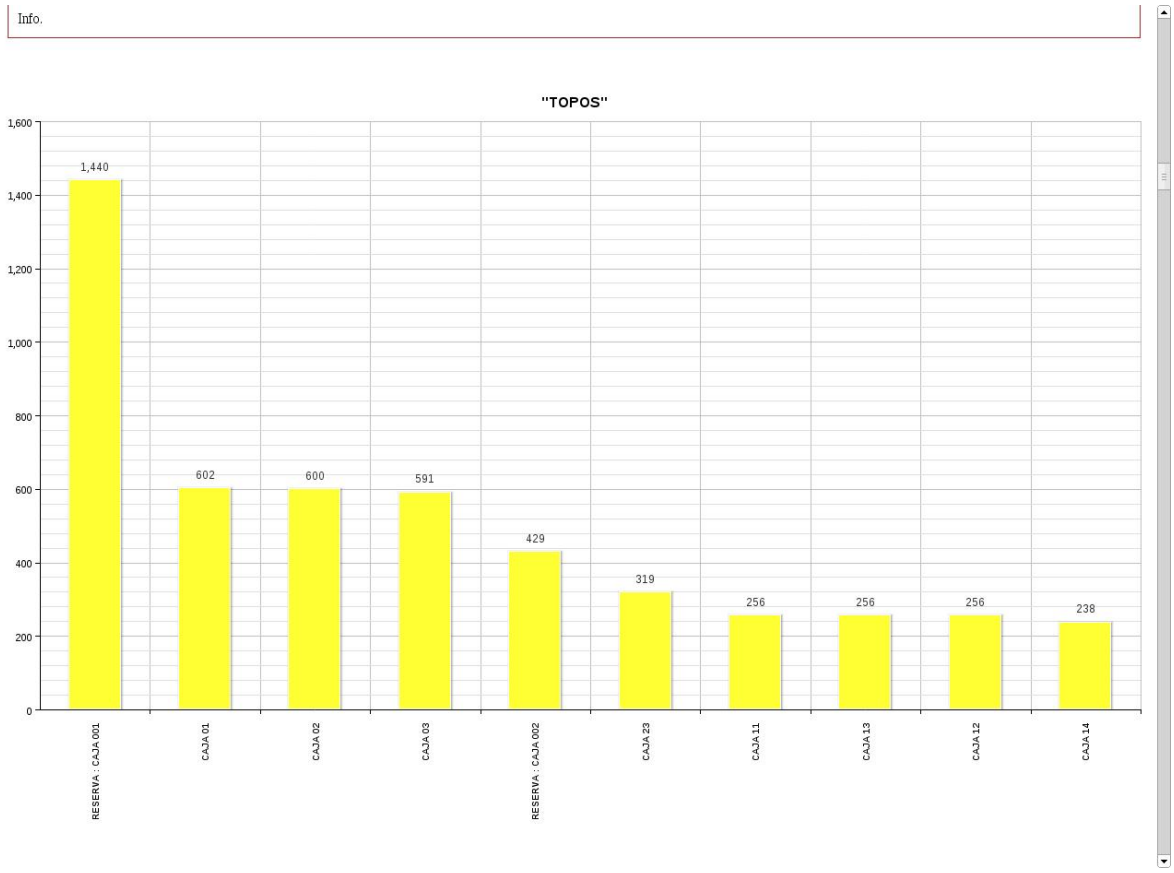

Figura 2.14: Gráfica de frecuencias para el modulo “Diagnósticos”

2.3.2. Funcionamiento

El proceso de este módulo inicia con el recurso “SetData” el cual recibe como parámetro un nombre, este nombre representa una estructura previamente configurada y dada de alta en el archivo de configuración “config_cfg” del cual se hablará más adelante. Como resultado se obtendrá una página HTML basada en la plantilla “template_diagnosticos” en donde se muestran las estadísticas en forma de gráficas.

El template cuenta únicamente con un dentro del body de HTML el cual tiene como valor “@id@” en el atributo “id”. Las @’s sirven para identificar los valores que serán sustituidos por el parámetro con el que fue llamado el recurso “SetData”. Lo que hace el recurso es únicamente obtener el contenido del template, sustituir la cadena “@id@” por el parámetro y regresar el template para que el navegador se encargue de codificar el código HTML.

Una vez que el navegador abre el template, se ejecuta el script “constructCharts” el cual se encarga de crear las gráficas en tiempo real. Para esta tarea utiliza el recurso “ConnectDB” el cual obtiene y formatea la información que presentan las gráficas.

El recurso “ConnectDB” recibe como parámetro el id del tag “div”, el cual fue sustituido por la cadena “@id@”. Esta cadena la utiliza para localizar el conjunto de acciones que debe realizar dentro del catálogo que se encuentra en el archivo de propiedades. Este archivo contiene un arreglo de JSON´s en donde cada uno de ellos representa un diagnóstico, es decir, en cada JSON se especifica la base, la tabla, las gráficas que se mostrarán y los campos que se contemplarán para la ejecución del diagnóstico.

El archivo de propiedades está construido de la siguiente forma:

```
[
    {
        “pathname” : “identificador”,
        “poolname” : “nombre del pool que realiza la conexión a la base de datos”,
        “graphtypes” : [“nonemptyfields”,”frequencyfields”],
        “nonemptyfields” : {información},
        “frequencyfields” : {información}
    },
    {
        …
    },
    …
]
```

Como se observa en la estructura, es un arreglo de JSON’s los cuales tienen la misma estructura interna. El valor de la etiqueta “pathname” es el que identifica a cada entrada del arreglo y son los que se utilizan como parámetro para iniciar el proceso y saber qué entrada ejecutar.

En “poolname” se especifica el pool de conexiones que se utilizará para realizar la conexión a la base datos y las consultas necesarias. Un pool de conexiones es un mecanismo que se encarga de administrar las conexiones a la base de datos, conexiones limitadas que pueden ser reutilizadas y es administrado por el servidor. Para el uso se requiere primero crearlo desde el servidor con la información necesaria para el acceso a la base de datos y una vez creado, únicamente nos encargaremos de consumirlo, el servidor será el responsable de crear la conexión, además de cerrar la sesión.

En la etiqueta “graphtypes” se almacena un arreglo en donde cada entrada indica el tipo de gráfica que se utilizará para el diagnóstico general, en estos momentos, solo están especificados dos tipos de gráficas, pero se pueden agregar más si los requerimientos aumentan y no es necesario solicitar todos los tipos.

Finalmente se crea una etiqueta por cada tipo de gráfica especificado en “graphtypes”, en ellas se almacena toda la información requerida para cada gráfica, tales como el nombre de la tabla, los campos de la tabla que se tomarán en cuenta dentro de la gráfica y las etiquetas de cada uno de ellos. En el caso de las gráficas de tipo “nonemptyfields” se especifican los campos que aparecerán en la gráfica de no nulos, en el tipo “frequencyfields” se especifican las columnas que se graficarán, esto es, se crea una gráfica por cada columna.

Una vez que se localiza el JSON correspondiente, se realizan las consultas necesarias a la base de datos para obtener la información, se almacenan los datos en una estructura válida para la biblioteca “Highcharts” y el resultado se envía como respuesta. Y para el final, el script únicamente obtiene los datos de la respuesta y ejecuta las funciones que crean las gráficas con los datos obtenidos. El proceso se muestra en la figura 2.15.

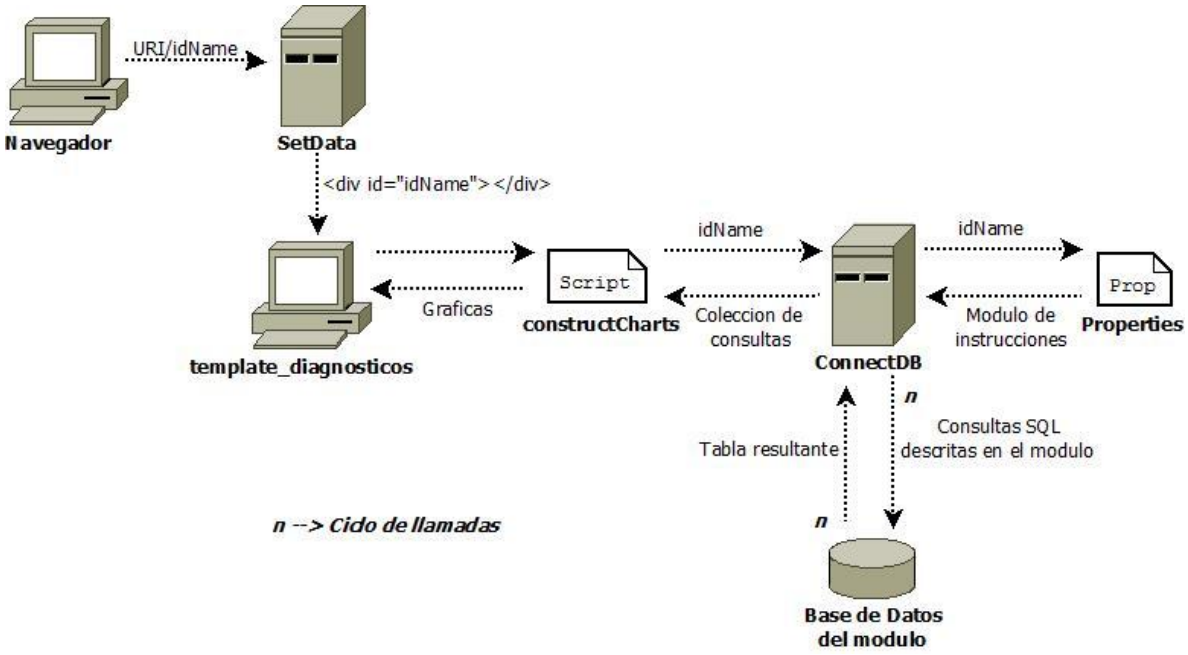

Figura 2.15: Proceso para modulo “Diagnósticos”.

2.3.3. Ventajas

Hasta ahora, sólo están implementadas dos tipos de gráficas, pero el módulo es escalable y se pueden agregar más tipos de gráficas sin alterar el funcionamiento actual. Funciona con cualquier base de datos sin necesidad de alterarla o agregarle alguna funcionalidad ni nada parecido. El diagnóstico puede ser lo más profundo que el usuario lo quiera, ya que tiene la capacidad de especificar las columnas de interés y estas pueden ser todas o algunas en los diferentes tipos de gráficas.

2.3.4. Desventajas

Este módulo requiere de diversas configuraciones previas para su funcionamiento, primero se debe de crear y configurar un pool de conexiones para la base de datos a la que se le realizará el diagnóstico, posteriormente, se debe de crear el JSON con todas las especificaciones, incluyendo los campos a valorar y si estos llegasen a ser bastantes, puede llegar a ser una tarea muy tediosa. Con todo esto, la adición de un nuevo diagnóstico, requiere de varios pasos y se requiere de conocimiento en la materia además de permisos ya que es necesario entrar al servidor.

2.3.5. Problemas

El módulo está hecho para diagnosticar alguna tabla de cualquier base de datos, no existe un límite en el tamaño de las tablas, al realizar pruebas con bases grandes, el proceso tardaba bastante, y la experiencia de usuario se veía afectada severamente. Para reducir el tiempo de espera se reestructuró el proceso y se generalizaron actividades para mejorar el rendimiento. Uno de los cambios más importantes fue el proceso de formatear los datos para crear las gráficas ya que en un principio se formateaban conforme se iban creando las gráficas, con el cambio, este proceso se generaliza y se lleva a cabo dentro del Servicio Web “ConnectDB”. Esto tiene como beneficio que el proceso se realiza a nivel servidor y muestra un mayor rendimiento.

3. Desarrollo de Servicios Web

Los Servicios Web desarrollados en la plataforma RESTful Java utilizando el framework Jersey (JAX-RS) están compuestos principalmente por anotaciones (annotations) y Java puro. Jersey realiza la conexión a los Servicios Web a través de las anotaciones de Java, las cuales generan automáticamente el código necesario para la interacción.

En general, un recurso Jersey es una clase Java la cual contiene contextos de una aplicación web como la anotación @Path que identifica al recurso dentro de la URI, y todos los recursos Jersey tienen una URI que apuntan a ellos.

Otras anotaciones son las que representan las funciones HTML: @GET, @POST, @PUT, @DELETE. Dentro de la definición de estas funciones se pueden agregar las anotaciones @Consumes y @Produces que definen el tipo de entrada o salida respectivamente, @Consumes está disponible solo para las funciones POST y PUT y @Produces para GET, POST y PUT.

Finalmente están las anotaciones que definen la forma en que se reciben los parámetros de cada función las cuales pueden ser: @FormParam que obtiene los parámetros a partir de una forma HTML, @PathParam la cual extrae los parámetros de la URI, @QueryParam también obtiene los parámetros de la URI pero limita estos con el caracter “?” y utiliza duplas “nombre=valor”, @DefaultValue contiene un valor predeterminado en caso de no recibir ningún parámetro. Esta anotación siempre va acompañada de otra anotación tal como @QueryParam, entre otras.

En este capítulo serán analizados tres proyectos para ejemplificar el desarrollo de los Servicios Web y algunas de sus capacidades, los cuales son:

- WSDataTools
- Queue
- WSImageMosaic

3.1. WSDataTools

Este proyecto pertenece a un conjunto de herramientas diseñadas para la limpieza de datos dentro de una base de datos, esta limpieza se basa en encontrar registros que pudieran ser lo mismo pero que aparecen como distintos valores. Aborda diferentes puntos como:

- Errores de escritura a la hora de la captura de registros, esto es; palabras que representan lo mismo pero varían en pocos caracteres, ya sea que les falten o tenga variantes. Por ejemplos:

  México – Mejico - Mexic

- En campos abiertos en donde se registran oraciones tienden a variar en el orden de las palabras y eso ocasiona que sean dos registros diferentes, cuando realmente dicen lo mismo pero en distinto orden. Esto tiende a mostrarse con frecuencia en la descripción geográfica de las colectas, ya que existen registros muy antiguos en los cuales no existían patrones para definir el orden de la descripción, estos pueden ser:

  + 4 km. al E de San José de Gracía, sobre el camino a Pabellón.

  + Sobre el camino a Pabellón, a 4 km. Al E de San José de Gracía.

- Otro punto que abarca la limpieza son los números, los cuales algunas veces se escriben con cifras y otras con letras y esto causa la duplicidad de los datos. Tal como:

  + Kilometro 57 – Kilometro cincuenta y siete

Este proyecto está compuesto por dos clases Java: “DBConnection.java” y “DataSourceAnalizer.java”. Así como un Servicio Web “Freqtable”.

3.1.1. Objetivo

Realizar un análisis de los registros de una tabla dentro de una base de datos a partir de tablas de frecuencia, las cuales muestran el número de veces que existe un registro dentro de una columna. Esto nos da una comparativa entre posibles registros que representan lo mismo, los registros con menor frecuencia, tienden a ser los erróneos, ya que es difícil cometer frecuentemente el mismo error a la hora de la captura. En general, nos da un panorama del estado de los registros de la base de datos y que tan limpia se encuentra.

3.1.2. Funcionamiento

El Servicio Web “Freqtable” recibe como parámetro la información necesaria para interactuar con las dos bases de datos requeridas, la base de datos objetivo y la base de datos destino. En la base objetivo se obtienen los registros en cuestión y en la base destino se crea la tabla de frecuencias con esos mismos registros (estas bases de datos pueden ser la misma).

Obtenidos los parámetros, se utiliza la clase “DBConnection”, la cual se encarga del manejo de la base de datos. Esta clase se ocupa de la conexión, realiza consultas, manipula los datos (agrega y quita registros) y cierra la conexión. Cuenta con un método “createTable” que obtiene una sub tabla ya sea a partir de una matriz de Java (arreglo de arreglos) o de un objeto ResultSet de Java, que es un objeto que representa una tabla de una base de datos.

Una vez que se realiza la conexión a la base de datos objetivo a partir de la clase “DBConnection” junto con los parámetros necesarios (dirección de la base de datos, usuario y contraseña), se obtiene la tabla a analizar, esta se logra a partir del nombre de alguna tabla de la base de datos y una lista con el nombre de las columnas de interés de la misma (en caso de que esta lista se encuentre vacía, se analizan todas las columnas).

Después, la clase “DataSourceAnalizer” es la encargada de conectarse a la base de datos destino y crear la tabla de frecuencias, esta se realiza a partir de la tabla obtenida de la base de datos objetivo, se verifica si la lista de columnas es vacía y en caso de no serlo, se obtiene una sub tabla únicamente con las columnas de la lista, esto se hace utilizando el método “createTable”. Posteriormente se recorre cada columna y se guardan todos los registros en una tabla temporal para después realizar un “GROUP BY” de SQL y obtener las frecuencias de cada registro. Al final la tabla de frecuencias se guarda en la base de datos destino y el Servicio Web “freqtable” regresa los datos necesarios para consultarla, la figura 3.1 muestra el proceso.

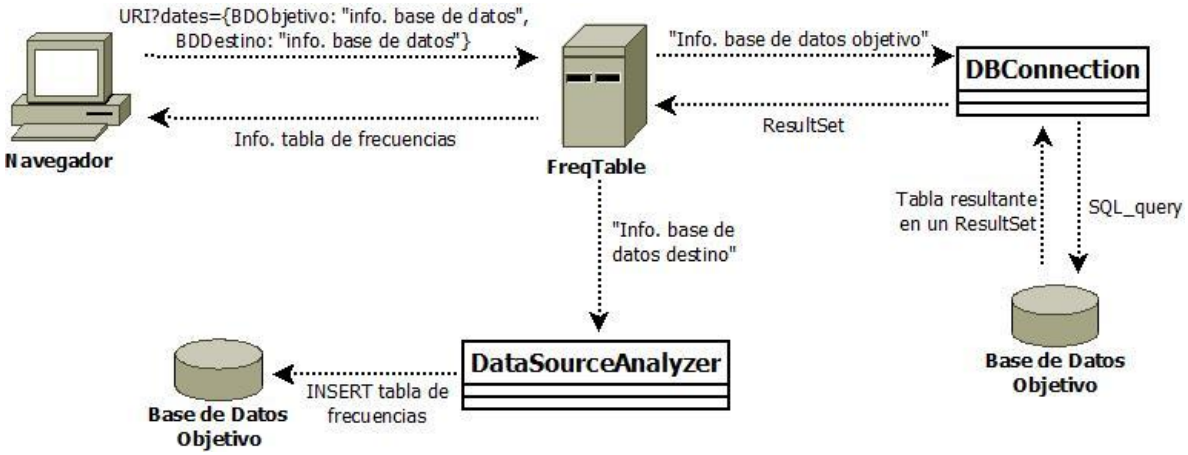

Figura 3.1: Proceso para la creación de las tablas de frecuencia para el modulo "WSDataTools"

.

3.1.3. Ventajas

Una de las principales ventajas que tiene este módulo es la de obtener la tabla de frecuencias de cualquier base de datos, además de manipular un gran número de registros sin temor a saturar la memoria, y esto es gracias a que los registros se van insertando en grupos de elementos y no todos al mismo tiempo.

3.1.4. Desventajas

Esta herramienta como tal, aporta poca información acerca del estado de una base de datos, en particular de una tabla, y por si sola puede ser poco funcional. Es una herramienta de apoyo a algunas otras herramientas como puede ser un unificador, el cual se encarga de buscar registros sintácticamente parecidos y a partir del número de frecuencias, decidir el correcto y si son iguales.

3.1.5. Problemas

Existe una clase Connection en Java, quien se encarga de conectarse a la base, y esta, requiere de otra clase llamada Statement, la cual se crea a partir de Connection y se ocupa de ejecutar comandos SQL (CREATE, SELECT, UPDATE, etc.). Estas clases son utilizadas en esta herramienta y forman parte de la biblioteca de Java "java.sql".

El problema surge cuando el método "createTable" obtiene un ResultSet con la tabla en cuestión y realiza modificaciones en la base de datos. Esto ocasionaba la pérdida de la información del ResultSet, ya que este depende del Statement y al hacer alguna modificación, se requería el Statement y se reiniciaba su valor. La solución fue inicializar dos Statements, una para las consultas y otro para la modificación de la base de datos.

3.2. WSImageMosaic

"WSImageMosaic" es un módulo que pertenece a la actualización de la página web de la UNIBIO, el cual da acceso y es un previo al portal Irekani, un repositorio de imágenes. Es un mosaico en donde se muestran fotografías de forma aleatoria que son extraídas del repositorio, contienen una pequeña descripción que también surge del repositorio. Está compuesto por 7 marcos en donde se muestra la información y estas cambian cada que se refresca la página.

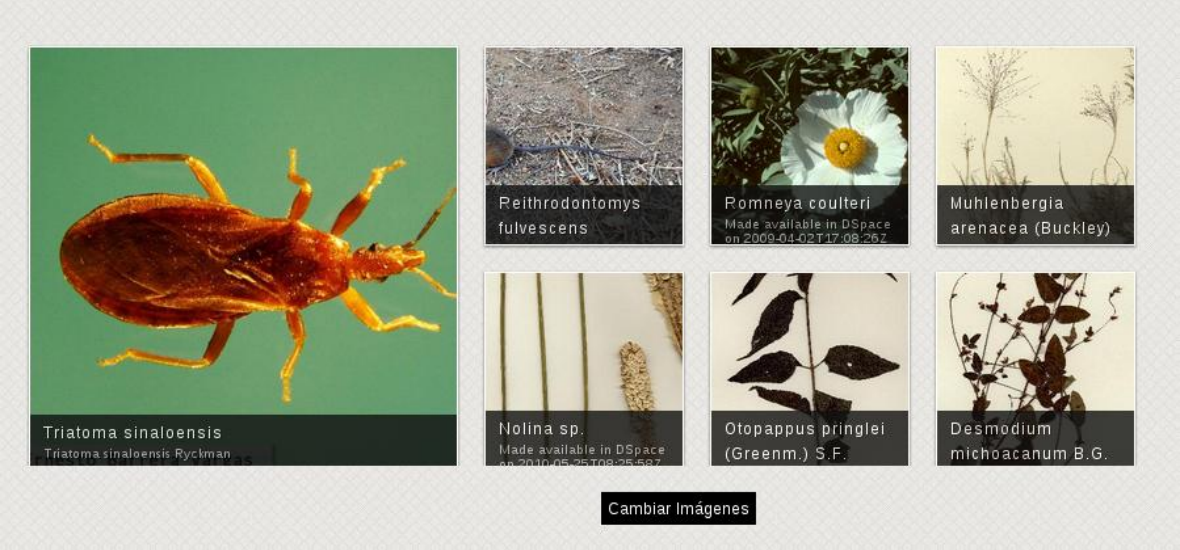

Figura 3.2 Mosaico de imágenes.

Está compuesto por un Servicio Web llamado "WSGetImages" quien obtiene 7 imágenes de forma aleatoria de la base de datos, esto lo hace a partir de la clase "RandomDates" la cual recolecta toda la información requerida para el módulo (URL de la imagen, URL de la ficha que pertenece a la imagen dentro del portal Irekani, además de una pequeña descripción).

La obtención de la información se realiza a partir de la clase "DBManager" la cual utiliza un archivo de propiedades "config_cfg.properties" que contiene toda la información para el funcionamiento de dicha clase y las consultas necesarias para cubrir el objetivo. Otra parte importante del módulo es el script "images" el cual se encarga de las imágenes, esto es: acomodar cada imagen dentro de cada marco del mosaico, además de mostrarla de forma centrada, tomando en cuenta que el tamaño de las imágenes no se encuentra estandarizado. Utiliza un par de plugins de jQuery: "mosaic! jQuery plugin" el

cual crea contenedores de imágenes con diversos efectos visuales, y “jCrop” que se encarga de obtener extractos de alguna imagen, el tamaño del extracto, entre otras cosas.

3.2.1. Objetivo

El objetivo de este módulo es la pre visualización del portal de Irekani de una forma atractiva y llamativa para el usuario, y funciona como enlace para el portal. En este se muestran pequeñas imágenes del portal y con ellas se pretende dar una idea de los tipos de fotos que se encuentran en el repositorio. Este módulo se realizó para el rediseño de la página de la UNIBIO, y se pretende colocar este mosaico en la sección del repositorio de imágenes.

3.2.2. Funcionamiento

Este módulo funciona de forma inmediata una vez que se accede a él, lo hace ejecutando el script “images”. La primer instrucción que realiza es la petición al recurso “WSGetImages” y comienza de forma inmediata ya que el recurso no requiere de ningún parámetro dado que siempre nos devuelve un arreglo de 7 elementos (JSON’s) los cuales contienen la información de cada imagen.

Una vez que se realiza la petición al recurso, este utiliza la clase “RandomDates” que se encarga de obtener los datos de forma aleatoria, quien a su vez utiliza la clase “DBManager” para realizar las consultas. Primeramente se obtiene el total de registros existentes en la base de datos a partir de una consulta, esto se hace contando las llaves de cada registro. Este resultado nos sirve para obtener un rango en el cual estarán nuestros números aleatorios, los cuales representan una posición dentro de la tabla de la base de datos y así regresar diferentes imágenes en cada petición, esto de forma aleatoria.

Teniendo el total de registros, se obtiene toda la información necesaria para el mosaico, se utiliza nuevamente la clase “DBManager” y la clase “Random” de la biblioteca de Java “java.util”, la cual nos da un número aleatorio y recibe como parámetro un número que es utilizado como límite (y es aquí donde se utiliza el total de registros para obtener siempre registros válidos). Este número se agrega en la consulta como valor para la cláusula OFFSET de PostgreSQL que funciona para indicar la posición del registro que deseamos, y va acompañada de la cláusula LIMIT, quien recibe también un número que indica la cantidad de registros que queremos. Finalmente almacenamos la información del registro en un JSON y este a su vez se almacena en un arreglo, este proceso se realiza para cada imagen (en este caso, el proceso se repite 7 veces que son el número de imágenes que se requiere), al final se obtiene un arreglo de JSON’s con longitud de 7 y con toda la información de cada imagen y es lo que devuelve el recurso “WSGetImages”, el proceso para la obtención de las imágenes se muestra en la figura 3.3.

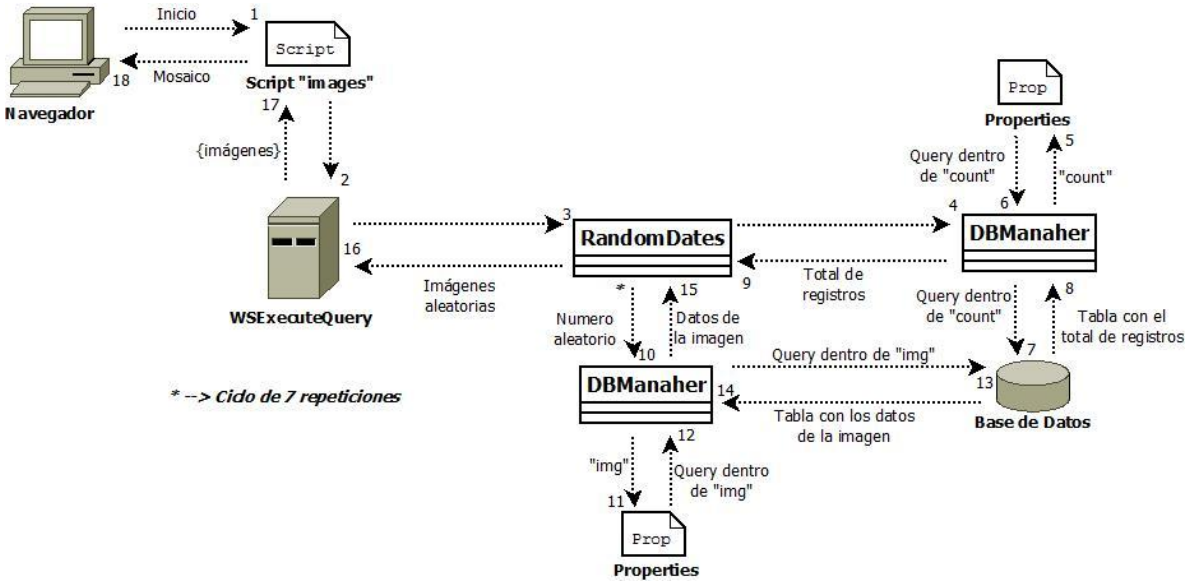

Figura 3.3: Proceso para la obtención de las imágenes dentro del modulo “WSImageMosaic”

Con toda la información necesaria de las imágenes obtenidas por el recurso, ahora toca manipularla para crear el mosaico y mostrarla al usuario de forma gráfica, y de esto se encarga el script “images” con la ayuda de los plugins “mosaic! jQuery plugin” y “jCrop”.

Primero son creados los mosaicos y son establecidos los efectos visuales, esto se realiza gracias al plugin “mosaic! jQuery plugin” y en el cual le es indicado a partir de parámetros el tipo de efectos que se quiere. También se debe de indicar la URL de la imagen y la descripción de ésta, además del enlace de cada imagen. Cada marco está representado por un “div” de html y en este se encuentra toda la información (enlace, imagen, título y descripción) y están establecidos por tags. En general son valores fijos pero gracias a jQuery y Ajax se pueden establecer los valores de estos a la hora de cargar la página, y así varían dependiendo de los resultados obtenidos del recurso.

“jCrop” es un plugin que se encarga de cortar las imágenes, de obtener una vista previa en menor tamaño de la imagen original o incluso una pequeña imagen de la original. Este módulo se utiliza para obtener un extracto centrado de las imágenes del repositorio y así mostrar en cada marco lo importante de cada imagen que regularmente se encuentra en el centro de la misma.

Pero las imágenes del repositorio son de diferente tamaño y no se puede establecer las coordenadas del extracto dentro de la imagen, estas coordenadas se tienen que calcular dependiendo de las medidas de la imagen original y el tamaño del marco,

realizando una función para cada marco que depende de la imagen que le toque, esta función obtiene las coordenadas del extracto y también edita los elementos CSS para el posicionamiento de esta. Después de obtener el extracto centrado correctamente, se muestra finalmente. Cada que se refresca la página, este proceso se repite y las imágenes son distintas.

3.2.3. Ventajas

El repositorio cuenta con fotografías de diversas colecciones y el hecho de que la selección de imágenes del mosaico sea de forma aleatoria, hace que se muestren fotografías diversificadas, además de ejemplificar el tipo de fotografías que contiene el repositorio.

El uso de plugins hace que el módulo sea vistoso y atractivo para los usuarios y el acceso a este sea mayor ya que fomenta la curiosidad de los usuarios además de no pasar desapercibido. Agrega además vista al portal que lo contenga.

Por ultimo la ventaja principal es la rapidez con la cual obtiene y organiza las imágenes de forma aleatoria.

3.2.4. Desventajas

Como se sabe, las imágenes que se muestran, se seleccionan de forma aleatoria y la visualización del mosaico siempre será diferente, si el usuario quisiera volver a visualizar una o varias de las imágenes ya visualizadas, esto será por decirlo de una forma, imposible y como buscador de imágenes resulta una herramienta inservible.

Otra desventaja son algunas imágenes del repositorio, las cuales son fotografías de etiquetas de ejemplares y no el ejemplar como tal, esto hace que de pronto salte a la vista del mosaico, letras, que no son visualmente atractivas.

3.2.5. Problemas

Uno de los principales problemas que se encuentra en el desarrollo de este módulo era a la hora de acomodar las imágenes en cada marco. Como se menciona anteriormente, cada imagen tiene dimensiones diferentes y los marcos son más pequeños y de tamaños fijos. Una opción era modificar el tamaño de las imágenes con el de los marcos, pero esto provocaría la deformidad de la imagen. Si las imágenes se presentaban tal y como eran obtenidas, se mostraba solo la parte superior izquierda y no se mostraba realmente el ejemplar que normalmente se encuentra en el centro de la imagen y es ahí donde se tuvo que buscar alguna herramienta para solucionarlo de forma rápida y eficaz, que fue a través de “jCrop”.

Como se dijo anteriormente, “jCrop” es un plugin de jQuery que obtiene un extracto de una imagen de cualquier tamaño y este muestra una parte de la imagen, en este caso era justo lo que se buscaba, mostrar un extracto de la imagen de su parte central, pero surge otro problema ya que “jCrop” funciona con dos imágenes, la original y la vista previa, que es el que se quiere mostrar realmente. Para solucionar esto, se almacenaron todas las imágenes originales dentro de un “div” y una vez que se obtenía el extracto de cada imagen, este “div” se esconde gracias a la función de jQuery “hide”.

Por último, uno de los problemas que surgieron fue el cálculo de los cuatro puntos del extracto de la imagen, posicionarlos para obtener el centro de la imagen. Para esto fue necesario obtener las medidas de cada imagen y a partir del tamaño del marco, calcular cada punto.

3.3. WSQueue

El Instituto de Biología de la UNAM alberga colecciones importantes a nivel nacional y de gran tamaño, estas sirven para la investigación, entre otras cosas. La UNIBIO se encarga de digitalizar esas colecciones además de crear herramientas web para facilitar el acceso, las consultas y la centralización de la información. Algunas herramientas tienen la opción de exportar la información a formatos comunes como pueden ser PDF o CSV. Los archivos CSV son de gran utilidad para los investigadores, ya que representan tablas organizadas con la información requerida y pueden ser visualizados en hojas de cálculo.

Como se manifestó con anterioridad, la información con la que cuenta el instituto es muy amplia lo que conlleva a que las consultas que se realizan de esta información, llegan a ser muy extensas y de gran tamaño, esto, a nivel servidor puede traer consecuencias graves y el servicio verse interrumpido, considerando el peor de los casos que sería una concurrencia de peticiones que son generalmente muy pesadas y tardías para la obtención del resultado. En particular, la exportación de archivos CSV tarda de forma proporcional al tamaño de la consulta y si se realizan varias consultas a la vez, el servidor puede saturarse y tardar más de lo normal o errores mayores.

La herramienta “WSQueue” fue pensada justo para solucionar la posible saturación del servidor y mantener la fluidez normal de la información. Está compuesta por dos Servicios Web: “QueueWS” y “WSConsult”, tres clases Java: “ClienteQueue”, “MDBQueue” y “CreateFile”, un archivo de propiedades “MailParameters” y una clase externa alojada en el servidor “MailSender”. Además se utiliza la API de Java: “JMS”.

3.3.1. Objetivo

El objetivo de esta herramienta es proteger la integridad del servidor y la disponibilidad del servicio para la exportación de archivos CSV. Esto lo logra administrando las peticiones de los usuarios de tal forma que ejecuta una a la vez, para obtener la mayor prioridad y el término del proceso sea más rápido. Para lograrlo, utiliza una fila en donde cada petición se almacena y el sistema va ejecutando cada una en el orden en el que se hayan almacenado (el primero que entra es el primero que se ejecuta). Al final, se le notifica al usuario final el estado de su petición por correo, ya sea que está lista para descargarse o que hubo algún error y que nuevamente realice la petición.

3.3.2. Funcionamiento

La API de Java "JMS" (Java Message Service) está enfocada al manejo de mensajes entre aplicaciones basadas en Java. Es un estándar de mensajería para que las aplicaciones puedan crear, enviar, leer y recibir mensajes y pueden ser de forma síncrona o asíncrona. En la forma asíncrona se utiliza una cola de tipo FIFO (First In First Out) en la cual se almacenan los mensajes. Existen dos tipos de modelos:

- *Modelo Publicador/Suscriptor.*- En este modelo existen varias aplicaciones que se encargan de consumir los mensajes.
- *Modelo Punto a Punto.*- En este modelo solo existen 2 clientes, uno que envía mensajes y otro que lo recibe.

En caso de que no esté disponible el/los consumidor(es) según sea el modelo, los mensajes se almacenan en la cola y son atendidos según hayan entrado.

Para la creación de un sistema de mensajes (JMS) se requiere la creación de dos estructuras básicas; un ConnectionFactory que se encarga de la administración de conexiones, y un Queue (cola), que es la estructura en la que se almacenan los mensajes. Estas, se crean a nivel servidor, esto es, el administrador es el encargado de crearlas.

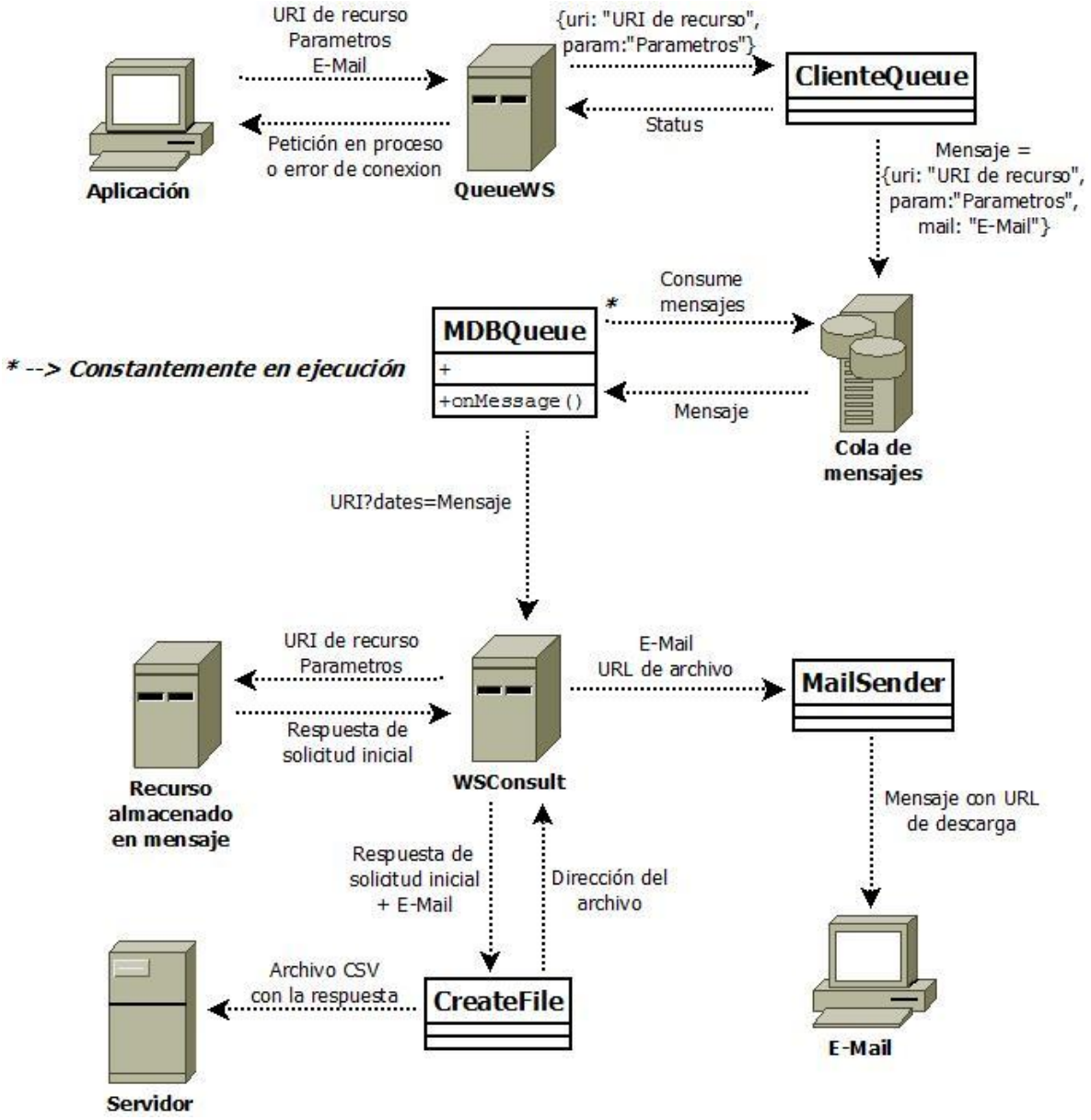

Figura 3.4: Proceso de encolamiento de procesos.

Analizando la figura 3.4, el proceso inicia con el recurso “QueueWS” el cual recibe como parámetros la URI y los parámetros del recurso a ejecutar, además de un correo electrónico. Se encarga de encolar como mensaje la URI junto con sus parámetros. En este caso particular, cada mensaje representa una petición.

Para el proceso de encolamiento del mensaje, “QueueWS” almacena los parámetros en un JSON quien recibe la clase “ClienteQueue”, que se encarga de realizar la conexión, establecer la cola a utilizar, crear una sesión y almacenar el mensaje en formato

de texto. Una vez que se almacenó la petición u ocurrió algún problema, el recurso "QueueWS" le notifica al usuario, ya sea que la petición está en proceso o que hubo problemas a la hora del proceso de encolamiento.

Una vez que los mensajes se encuentran almacenados en la cola, el receptor se encarga de consumirlos, en este caso "MDBQueue", hace referencia a la cola en cuestión y se encuentra activo en todo momento verificando si existe algún mensaje en la cola. Si consume alguno, al término del proceso, verifica si no existe algún otro. El método "onMessage" es el que se encarga de procesar cada mensaje, el procesamiento de estos consiste únicamente en la invocación del recurso "WSConsult" con el mensaje como parámetro.

Finalmente el proceso termina en el recurso "WSConsult" pero es en donde se realizan las tareas más importantes, que tienen que ver con la obtención del resultado de la petición del usuario, además de la notificación por correo electrónico del estado de esta. Primero se manipula el mensaje para así alcanzar los parámetros iniciales, se obtiene un JSON con la información, una vez que se pueden acceder a ellos con facilidad, se realiza la petición de los parámetros y se tiene la respuesta.

Lo que sigue es almacenar la respuesta en un documento CSV en el servidor, esto se realiza con la clase "CreateFile" que es quien recibe la respuesta y el correo electrónico que junto con la hora y fecha, crean un nombre único para el archivo. Primero crea el archivo, posteriormente el contenido de este a partir de la respuesta y al final regresa el nombre del archivo. Estos archivos son almacenados en el servidor, dentro de una carpeta con los permisos necesarios para que cualquiera sea capaz de descargarlos.

Por último, con el nombre del archivo, se crea una URL para la descarga del archivo y se manda el correo electrónico con esta URL. Esta tarea la realiza la clase "MailSender" la cual se ejecuta desde el servidor y se invoca desde el recurso "WSConsult" junto con el correo electrónico y el contenido del mensaje a enviar (en este mensaje se encuentra la URL del archivo en caso de que no haya existido ningún percance, de lo contrario, el mensaje es un aviso que notifica al usuario que ocurrió un error en el proceso). Esta clase utiliza la biblioteca de Java "javax.mail" para la conexión, creación y envío del correo electrónico.

3.3.3. Ventajas

Una de las principales ventajas es el control de la concurrencia, ya que la maneja de una forma muy eficiente y evita la caída del servidor por esta causa.

Otra ventaja es la notificación por correo electrónico a los usuarios, ya que no es necesario que el usuario esté esperando en el portal a que se genere su petición, esto le da libertad de hacer otras tareas al usuario mientras se obtiene la consulta.

3.3.4. Desventajas

Una desventaja sería la posible demora de la respuesta, ya que puede llegar a tardar bastante tiempo y depende de que tantas solicitudes se hayan realizado anteriormente, esto apunta directamente a que se atiende una sola petición a la vez y puede provocar una cola muy extensa.

3.3.5. Problemas

Uno de los principales problemas que se presentaron dentro del desarrollo de este módulo, era aprender una nueva tecnología ya que era la primera vez que se utilizaba y tenía que adecuarse a los Servicio Web dado que no se encontró un ejemplo en el que usaran recursos para la ejecución de mensajes.

Otro problema importante fueron las notificaciones por correo electrónico pues la configuración del servidor se tuvo que modificar actualizando un archivo de configuración, ya que en su versión anterior, la biblioteca “javax.mail” no funcionaba correctamente. Además de crear y utilizar el módulo que realiza esta tarea de forma remota, puesto que es una herramienta reusable y genérica. Esta se ejecuta desde el recurso “WSConsult” y está alojada en el servidor como un programa de Java (.jar).

4. Conversión de una herramienta a un Servicio Web

El objetivo principal de un Servicio Web es dar un servicio a través de la web, y para esto debe de tener ciertas características como ya ha sido analizado. Y aquí surge la pregunta: ¿Qué tan viable es hacer que el servicio que da una herramienta, pueda estar disponible en la web?

Puede haber dos opciones: una sería desarrollar la misma herramienta en algún lenguaje de programación interpretado (PHP, Javascript, etcétera) o alguna herramienta web (Servlets, JSP, etcétera), pero esto implica invertir tiempo en hacer algo que ya está hecho. La segunda opción, como es de imaginarse, sería convertirlo en un Servicio Web con todas las ventajas que esto conlleva. Al decir convertirlo se refiere a crear un Servicio Web el cual ofrecerá el mismo servicio que la herramienta en cuestión.

Para la conversión basta con adecuar el Servicio Web para que solicite los parámetros necesarios (en caso de que se requieran) y ejecutar la herramienta desde el Servicio Web. En general, toda herramienta que pueda ser ejecutada desde el lenguaje Java, puede convertirse en un Servicio Web, desde una aplicación Java usándola como biblioteca, hasta un script, utilizando bibliotecas especiales y líneas de comando.

En este capítulo se analizarán dos proyectos que utilizan Servicios Web y están basados en clases comunes de Java y que en un principio no se contemplaban como Servicios Web:

- PruebaFonetica
- WSUnifica

4.1. PruebaFonetica

Este módulo pertenece al conjunto de herramientas que se encargan de la limpieza de datos en las bases de datos. Está enfocado en la comparación fonética de dos palabras a partir de un diccionario fonético, este es llenado por el usuario y puede ser tan extenso como el mismo quiera, la estructura de este consiste en "fonema - valor numérico" y junto con un margen de error, el sistema decide si dos palabras son fonéticamente equivalentes.

Está compuesto por dos Servicios Web “PhoneticDictionary” y “Compare”; utiliza la herramienta “Soundex” la cual está compuesta por dos clases Java: “Gramatica” y “SoundexExt”.

4.1.1. Objetivo

El objetivo principal de esta herramienta es decidir si dos palabras son fonéticamente equivalentes, esto es, cuando se pronuncian, se emite el mismo sonido sin importar como se escriban. Para este proceso se utiliza el algoritmo “Soundex”, el cual indexa las palabras en cuestión, pueden ser silabas o solo parejas de consonante-vocal o viceversa, inclusive a una sola consonante se le puede dar un valor. Todo esto depende de las reglas que se ocupen para la ejecución del algoritmo. Una vez que se le proporcionan valores a las palabras a comparar, se obtiene la diferencia entre cada valor que se encuentren en la misma posición y se suman esos valores. A partir de un valor de error se decide si son o no equivalentes, si es menor o igual al error, son equivalentes, si es mayor no lo son. Esta herramienta se ocupa para identificar faltas de ortografía o errores de dedo en los registros de una base de datos.

4.1.2. Funcionamiento

Esta herramienta se divide en dos recursos, el primero está enfocado a la creación o agregación de la gramática, la cual es una tabla de dos columnas, en la primera se encuentra el fonema y en la segunda el valor para la indexación. El Servicio Web que se encarga de esto es “PhoneticDictionary”, recibe como parámetro la información para la conexión a la base de datos y a la tabla de la gramática, también recibe de 1 a n JSON’s que contienen las parejas que representan la gramática, esto es, en una sola petición se pueden agregar de 1 a n parejas (gramáticas).

Lo primero que realiza este recurso es convertir cada JSON que recibe, en objetos de la clase “Gramatica”, los cuales tienen como atributos una expresión y un valor. Posteriormente son almacenados en un arreglo para finalmente, utilizando la información de la base de datos, almacenarlos en la tabla señalada por el usuario.

Para almacenar el arreglo bidimensional dentro de la base de datos, se utiliza la clase “DBConnection”. En caso de que la tabla no exista, esta se crea como una nueva tabla y se almacenan de igual forma los datos. En caso de no haber ningún problema, se notifica el éxito del proceso, en caso contrario, se notifica el problema (figura 4.1).

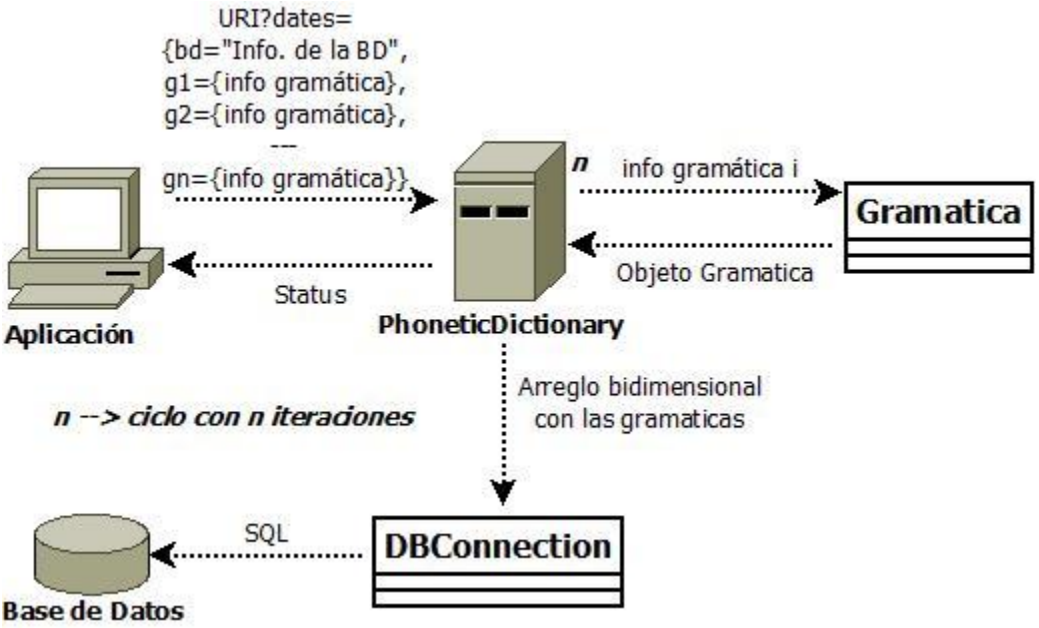

Figura 4.1: Proceso para la creación o actualización de un diccionario.

El segundo recurso llamado “Compare” es el encargado de la comparación de palabras a partir del algoritmo “Soundex” basado en la clase de Java “SoundexExt”, quien utiliza un objeto de la clase “Gramatica” para la indexación de las palabras y la decisión de equivalencia entre dos palabras.

Recibe como parámetros los datos necesarios de la tabla en donde se encuentran los datos para la construcción de la gramática, las dos palabras a comparar y finalmente un valor que representa el margen de error para la toma de decisiones.

Primero se obtiene los datos de la base de datos y los procesa para crear la gramática, posteriormente transforma las palabras en un conjunto de valores.

Se comparan los dos conjuntos de valores, uno a uno y se obtiene la diferencia de cada pareja de valores, en caso de que un conjunto tenga más elementos, se compara con 0 como valor. Una vez teniendo las diferencias, estas son sumadas y se comparan con el valor que representa el margen de error, si este es menor o igual al error, son equivalentes pero si es mayor, no son fonéticamente equivalentes. Finalmente, el recurso devuelve el valor booleano con la respuesta de equivalencia, además de los conjuntos de valores que corresponden a las palabras en cuestión (figura 4.2).

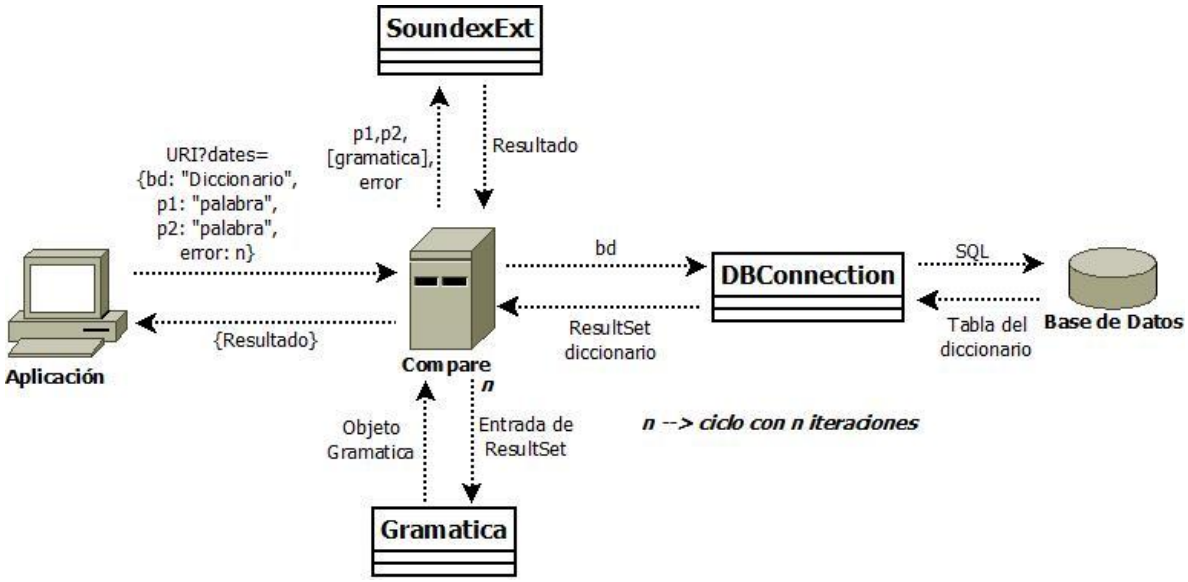

Figura 4.2: Proceso para la comparación fonética entre dos palabras.

4.1.3. Ventajas

Una de las ventajas de esta herramienta es que el usuario puede crear e incrementar la gramática tanto como él quiera y entre más grande sea la gramática, mayor número de palabras pueden ser comparadas además de mejorar la precisión en la comparación de las mismas. Otra ventaja que tiene es la adición de reglas para la gramática ya que se pueden agregar n reglas en una sola petición. Finalmente, se puede tener diversas gramáticas en diferentes partes y pueden ser consultadas de la misma forma, únicamente se requieren los datos necesarios para establecer la conexión a la base de datos.

4.1.4. Desventajas

Las peticiones para ambos recursos pueden llegar a ser un poco engorrosas, puesto que se deben de incluir todos los datos de la base de datos en donde se encuentra la tabla que representa la gramática. Otra desventaja puede ser la falta de una gramática genérica, ya que para realizar la comparación entre dos palabras, se requiere crear previamente una gramática que mínimo contenga los fonemas que componen las palabras en cuestión, y la creación de una gramática amplia y funcional puede ser un trabajo arduo de no ser que ya exista está en otro formato.

4.1.5. Problemas

Como ya fue mencionado, este módulo está basado en la clase Java “Soundex” realizada en la UNIBIO con anterioridad y sin la claridad de su funcionamiento y aplicación. Se tuvieron que realizar cambios de la estructura básica de la clase, así como aumentar lo necesario para poder convertirla en un Servicio Web. Entre uno de esos cambios así como también inclusión, fue el uso de bases de datos para la representación de la gramática. Para todo esto fue necesario analizar y entender el código del creador y así moldearlo para cubrir las necesidades requeridas.

4.2. WSUnifica

Este proyecto pertenece al conjunto de herramientas que se encargan de la limpieza de datos. Es un portal que se conecta a alguna base de datos y se enfoca en los registros de una determinada columna de alguna tabla. Utiliza como herramienta un diccionario, el cual está representado como una tabla. El proceso corre registro por registro y de cada uno, obtiene tres posibles opciones del diccionario y es el usuario quien elige la opción correcta. Una vez seleccionada la opción más adecuada, puede continuar seleccionando las opciones del siguiente registro o finalizar con los ya seleccionados, lo cual implica que el sistema almacena las respuestas seleccionadas en una tabla de resultados. Cabe señalar que el usuario debe de ser un experto en la materia, el sistema no es capaz de tomar decisiones por sí mismo.

Este sistema se basó en el proyecto de Java “Unifica”, la cual se encarga de obtener posibles equivalencias de oraciones, esto lo hace a partir de probabilidades, utilizando el teorema de Bayes.

Este sistema está compuesto de dos recursos “Sorter” y “Selections”, el proyecto de Java “Unifica”, a su vez contiene tres principales clases: “Tabla”, “Diccionario” y “ClasificadorBayes” que son las que se encargan de la funcionalidad principal, además se encuentran las clases “CrearTablaBayesTextos” y “ClasificaRegistrosBaseDeDatos” que son las que dotan al proyecto de la funcionalidad en las Bases de Datos.

4.2.1. Objetivo

El objetivo de este sistema es la limpieza de bases de datos, principalmente en registros de tipo texto, los cuales son abiertos y pueden ser descripciones que están compuestas por un conjunto extenso de palabras.

Esta limpieza se basa en obtener una lista de registros unificados, eliminar la concurrencia, encontrar posibles equivalencias en esas descripciones; describir lo mismo pero de forma diferente, que las oraciones estén en un orden diferente o que los números estén escritos en algunos registros con letras y otros con números, entre otros.

El sistema proporciona posibles sinónimos, los cuales cumplen estándares establecidos por un experto en la materia, o se tiene la certeza de que están correctamente escritas, el usuario experto decide si es alguna de las opciones proporcionadas, en caso de que no sea ninguna de las proporcionadas, se almacena el registro tal cual. Al final se obtiene una tabla con las selecciones que realizó el usuario.

El teorema de Bayes es un teorema que se basa en la probabilidad condicional, la cual está definida por la probabilidad de que ocurra un evento A sabiendo que ocurre el evento B. Lo que hace el teorema es relacionar la probabilidad de A dado B con la probabilidad de B dado A. Para ejemplificar el teorema, consideremos que se sabe la probabilidad de que una persona se resbale dado que está lloviendo, podríamos saber la probabilidad de que llueve si una persona resbaló. Para llegar a este resultado se requieren algunos otros datos.

En general, el teorema funciona para un conjunto de eventos $\{A_1,A_2,\ldots,A_n\}$ los cuales son mutuamente excluyentes y su probabilidad es distinta de cero. Sea B un evento del que se conoce la probabilidad condicional B dado A_i representada como:

$P(B|A_i)$ entonces $P(A_i|B)$ está dada por:

En este sistema se utiliza el teorema de Bayes, en donde los eventos A_i son las posibilidades que arroja el sistema y el evento B es el registro en cuestión.

4.2.2. Funcionamiento

El sistema cuenta con una interfaz inicial en donde el usuario inserta los datos requeridos para iniciar el análisis, estos datos son: URL de la base de datos, nombre y contraseña, nombre de la tabla que representa el diccionario, nombre de la tabla a analizar además de la columna de interés perteneciente a está (únicamente se analiza esta columna) y finalmente el nombre de la tabla en donde se almacenarán los resultados, esta tabla se compone de dos columnas, en la primera se almacena el registro y en la otra la opción seleccionada. El proceso se muestra a grandes rasgos en la figura 4.3.

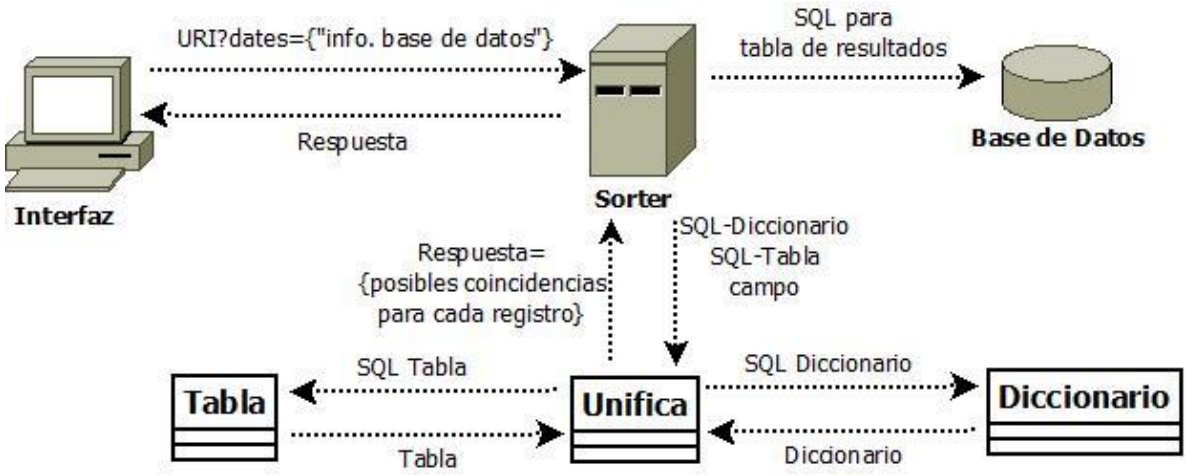

Figura 4.3: Proceso para obtener las tres coincidencias con mayor probabilidad.

Una vez que el usuario ha proporcionado los datos, son recolectados y almacenados en un JSON, el cual se envía como parámetro al recurso “Sorter”. Este recurso se encarga de crear la conexión a la base de datos para crear la tabla en donde se almacenarán los resultados del análisis, esto en caso de que no exista, si esta ya existe, simplemente se utiliza la existente. Si no hay percances en las acciones anteriores, se crea una instancia de la clase “Unifica” y en este punto se inicia el análisis de los registros.

El análisis inicia con el método “clasificaRegistros” el cual recibe dos cadenas que representan consultas SQL, la primera para la tabla que representa al diccionario y la segunda, los registros que serán analizados, además recibe dos banderas (números) que serán analizadas más adelante. Las consultas son creadas a partir de los parámetros que recibe (nombre de la tabla del diccionario, nombre de la tabla a analizar y el nombre de la columna).

Para el análisis se utilizan dos objetos principales: el primero es de la clase “Tabla”, los objetos de esta clase están compuestos por dos listas, una de palabras y otra de números de tipo Double, los cuales representan probabilidades. Estas dos listas son del mismo tamaño, además que están vinculadas de acuerdo a su posición de las entradas de cada lista, esto es: la probabilidad de la palabra que se encuentra en la posición i de la lista de palabras está representada en la posición i de la lista de probabilidades.

El segundo objeto es de la clase “Diccionario”, está compuesto por un objeto de la clase “Tabla” y una lista en donde cada entrada es una lista de palabras. En el objeto de tipo “Tabla” se almacena una palabra (identificador) y su probabilidad y en la lista de listas de palabras se almacenan las definiciones.

Una vez que se tienen las consultas SQL, se ejecutan para obtener los datos, teniéndolos, se crean los respectivos objetos. En el caso del diccionario, la consulta SQL nos devuelve una tabla con dos columnas, la primera se utiliza como identificador y en la segunda se obtiene la definición.

Se recorre la tabla para crear el objeto “Diccionario”, cada identificador se almacena pasando por un conjunto de filtros que eliminan caracteres que alteren los resultados o que no son funcionales (signos de puntuación, caracteres no alfanuméricos, espacios dobles). Las definiciones pasan por los mismos filtros, una vez aplicado el filtro, se crean listas con todas las palabras de cada definición y se almacenan. Las probabilidades del objeto “Diccionario” se obtienen a partir de otro objeto “Tabla” (probabilidad condicional).

Para la creación del objeto “Tabla”, la consulta SQL nos regresa una tabla con una sola columna, en donde se encuentran los registros que se analizarán, para cada uno de estos se va a crear un objeto “Tabla”. El registro pasa por los mismos filtros antes mencionados, después se obtiene una lista con todas las palabras, posteriormente se almacenan en la lista de palabras del objeto “Tabla” quitando repeticiones, finalmente se llena la lista de probabilidades la cual se calcula a partir de la frecuencia en que se encuentra cada palabra dentro de la lista original.

Una vez obtenida la frecuencia, la palabra se divide entre el número de elementos que tiene la lista de palabras del objeto “Tabla”. Ya teniendo los objetos necesarios se puede comenzar el proceso de análisis.

Como se dijo antes, las probabilidades del diccionario, dependen de otra tabla y se devolverán las tres entradas del diccionario con mayor probabilidad. Para obtener el valor de cada entrada del diccionario, se verifican las coincidencias de palabras, esto es, se busca cada palabra de la tabla en la definición del diccionario, si se encuentra dentro de la definición se suma 1, si se encuentra n veces, se suman n y si no se encuentra, se continúa a la siguiente palabra.

Cuando se finalizan todas las palabras de la tabla, el resultado es dividido entre el total de palabras que hay en todas las definiciones, y finalmente, esa división nos da la probabilidad de una entrada del diccionario, este proceso se realiza para cada una de las entradas de este. Por último cada probabilidad obtenida se multiplica por el producto de todas las probabilidades de la tabla.

Una vez que se tienen las probabilidades del diccionario a partir de una tabla, se almacenan las tres entradas del diccionario con mayor probabilidad. Este proceso se realiza para cada registro de la tabla obtenida por la consulta SQL. Las banderas

mencionadas previamente, funcionan para tener un control del proceso de selección, ya que una nos ayuda a realizar el análisis desde un registro n y la otra representa el número de registros a analizar. Con la ayuda de estas banderas, el proceso no se realiza en todos los registros de la tabla pues puede ser muy extensa.

Finalmente se formatean los resultados obtenidos y son enviados como respuesta. El cliente manipula estos resultados y muestra una pantalla de selección en donde el usuario selecciona la mejor opción de cada registro.

En esta interfaz se encuentran algunos botones y cajas con los que el usuario puede interactuar, uno de ellos es una lista con cantidades que representan el número de registros que se analizarán por cada consulta; un botón de siguiente, el cual al ser presionado, ejecuta una nueva consulta al recurso "Sorter" que nos regresará *n* resultados más a partir del último registro analizado; el botón de atrás realiza la misma acción que el botón siguiente pero con parámetros distintos ya que nos regresa los n registros anteriores; por último el botón guardar hace una llamada al recurso "Selections" el cual recibe como parámetros los datos necesarios para la conexión a la base de datos junto con el nombre de la tabla en donde se almacenarán los resultados de la selección del usuario.

Estos resultados están compuestos por el registro original y la selección del usuario para ese registro. Este recurso lo único que hace es almacenar los datos en la tabla proporcionada por el usuario y nos devuelve una notificación del estado del proceso (figura 4.4).

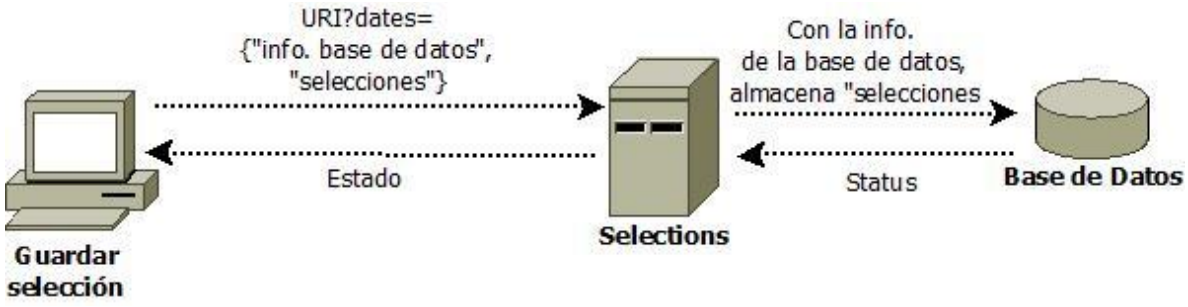

Figura 4.4: Proceso de almacenamiento para las selecciones del usuario.

4.2.3. Ventajas

Una de las ventajas del sistema es lo fácil de usar para el usuario final además de que se almacenan las selecciones del usuario y esto puede ser en cualquier momento que decida el usuario (desde una sola selección, hasta *n*). El proceso es rápido ya que no

realiza el análisis de todos los registros de la tabla, únicamente el número que indica el usuario. Se puede continuar el proceso en donde se quedó el usuario sin necesidad de volver a seleccionar lo ya seleccionado. Finalmente no es necesario realizar una selección de todos los registros, únicamente se almacenan los que seleccionó el usuario.

4.2.4. Desventajas

Una de las desventajas más notables es que el usuario debe de proporcionar toda la información para la conexión a la base de datos y esto puede ser muy engorroso además de la facilidad de errores en la información. El análisis solo se puede realizar en una sola columna de la tabla por sesión y no en varias. Finalmente los recursos deben de cargar información extra como los datos de conexión a la base de datos y las banderas necesarias para la continuidad de la sesión. Esto hace que las respuestas sean más complejas y la elaboración de estas sea más tardía.

4.2.5. Problemas

La clase “Unifica” fue realiza en la UNIBIO con anterioridad y en un principio no fue pensada para Servicios Web, esto obligó a modificar el código para así, cubrir los requerimientos.

Otra situación fue la persistencia del proceso, ya que como se sabe, los Servicios Web no la tienen, del lado del cliente es muy costoso el mantenimiento de variables globales, y esto se da ya que para cada consulta a los recursos, se realiza una conexión a la base de datos y se requiere la información necesaria. Para resolver esto, esa información se almacena en cada respuesta de los recursos y estos son usados por el cliente para realizar otra petición.

Conclusión

En concreto, un Servicio Web es un recurso que realiza una tarea particular, se encuentra alojado en alguna red e intercambia información con alguna otra aplicación. Utiliza la estructura cliente/servidor y está disponible en todo momento para cualquier cliente de cualquier tipo. Para consumirlo se requiere de protocolos comunes, los cuales funcionan como transporte para el intercambio de datos.

Dentro de un sistema, podemos ver a los Servicios Web como cajas negras, de las cuales, como clientes, solo se necesita saber la forma en la que se invocan y el resultado que se obtiene de la ejecución, la forma en cómo obtiene la respuesta y el proceso que utiliza se desconoce. Estas cajas comúnmente tienen la propiedad de ser genéricas, rehusarse el mayor número de veces, con esto se pueden abarcar distintos requerimientos y así reducir considerablemente el código en un sistema.

Hoy en día existen muchos sitios que tienen disponibles diversos Servicios Web, un claro ejemplo es Google y su servicio de mapas, los cuales usan Servicios Web para generar mapas con puntos, rutas o áreas dentro de algún portal. Lo único que debemos de saber para utilizar los mapas es la dirección URL del servicio y tener conocimiento en sus posibles parámetros.

En conclusión, los Servicios web son una herramienta que dotan de funcionalidades a un sistema además de cubrir requerimientos. Nos ayudan a reducir la tarea de programación además de poder utilizarse y estar disponible para el uso en diferentes sistemas, son muy fáciles de usar, de desarrollar y adaptarlos en otras herramientas.

Esta tecnología ayuda a la gestión de la información dentro de un sistema, manteniéndolo actualizado sin necesidad de modificarlo y proporcionando únicamente los datos necesarios para cada petición del usuario. Con estas características, obtenemos un sistema dinámico y ligero el cual está desarrollado con una cantidad de líneas de código menor. En definitiva, son herramientas fáciles de utilizar y con grandes ventajas, que si se desarrollan dentro de una empresa, se puede tener un gran ecosistema genérico que optimice cada vez más los sistemas que se realicen.

Este documento surge a partir del trabajo que se realiza dentro de la UNIBIO en donde los Servicios Web son una herramienta primordial y en donde se ha tenido la oportunidad de trabajar en todos los niveles de desarrollo, desde la creación de éstos,

hasta su consumo. Debido a esto es que decido realizar un documento que registre esta tecnología y mi experiencia en este tiempo que he trabajo dentro de la unidad.

El crecimiento que he tenido laboralmente dentro de la UNIBIO ha sido gracias a los conocimientos que adquirí en la carrera de Ciencias de la Computación, desde las materias de “ICC” (Introducción a Ciencias de la Computación) en donde aprendí las bases para programar en el lenguaje Java, la materia de “Lenguajes de Programación” en donde obtuve la habilidad de comprender y aprender nuevos lenguajes de programación, hasta las buenas prácticas que obtuve en la materia “Ingeniería de Software”

Al final me considero capaz de aprender con facilidad nuevas tecnologías, resolver problemas y requerimientos computacionales de cualquier empresa o institución de una forma óptima y correcta, trabajar en equipo y bajo presión. Todo esto gracias a las bases que me dio la carrera y la experiencia adquirida en la UNIBIO.

Bibliografía

.

SANDOVAL, José. RESTful Java Web Services: Packt Publishing Ltd. 2009. 258 p. ISBN: 978-1-847196-46-0.

"Guía breve de Servicios Web", [en línea]. Junio 2010, [19 de Febrero 2012]. Disponible en la web: http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb

"Representational State Transfer", [en línea]. [24 de Febrero 2012]. Disponible en la web: http://es.wikipedia.org/wiki/Representational_State_Transfer

"Web Services, un ejemplo práctico", [en línea]. [29 de febrero 2012]. Disponible en la web: http://msdn.microsoft.com/es-es/library/bb972248.aspx

BOTELLO CASTILLO, Alejandro. "Construcción de Servicios Web con SOAP", [en línea]. Marzo 2002, [24 de Noviembre 2012]. Disponible en la Web: http://www.revista.unam.mx/vol.3/num1/art3/index.html

RAMIREZ, Edgar. "SOAP", [en línea]. Octubre 2008, [20 de Noviembre 2012]. Disponible en la web: http://edgarramirez.wordpress.com/2008/10/14/soa/