

INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Sistemas Distribuidos

Proyecto Omega

Data Web Wizard

Reporte de implementación



Integrantes:

Victor Daniel Cruz González

Paola Mejia Domenzain

Fecha de entrega:

17 de mayo del 2019

Índice

1. Introducción y descripción del proyecto	3
1.1. Introducción	3
1.2. Descripción del proyecto	3
2. Documentación UML	3
2.1. Diagramas UML de clases	5
2.2. Diagramas jerárquico del proyecto	6
2.3. Diagrama de secuencia	7
2.4. Diagrama de bases de datos	8
3. Descripción de los obstáculos encontrados y cómo fueron re-	
sueltos	9
3.1. Iniciar sesión mediante un Servlet	9
3.2. Desplegar los servicios web	9
3.3. Actualizar SOAP	9
3.4. Parámetros en servicios Web RESTful	10
3.5. Cargar el JavaScript	10
4. Lista de Bugs conocidos	10
5. Conclusion	11

1. Introducción y descripción del proyecto

1.1. Introducción

El presente trabajo tiene como objetivo reportar la implementación del “Proyecto Omega”. A continuación, se encuentra una detallada descripción del diseño e implementación, incluyendo un diagrama de secuencia, diagramas de bases de datos, obstáculos y bugs.

1.2. Descripción del proyecto

El proyecto “Data Web Wizard” tiene el objetivo de manejar bases de datos utilizando servicios web SOAP accedidos desde servicios web RESTful. Asimismo, el proyecto refuerza el manejo de sesiones utilizando HttpSession.

A grandes rasgos, un usuario podrá registrarse y crear tablas en una base de datos. Las tablas del usuario podrán modificarse o ser consultadas solamente por el usuario.

2. Documentación UML

A continuación, se presentan los diagramas UML de las clases principales que son Conexion, DataBaseService, OperationResource. Los diagramas fueron generados con la extensión de NetBeans easyUML.

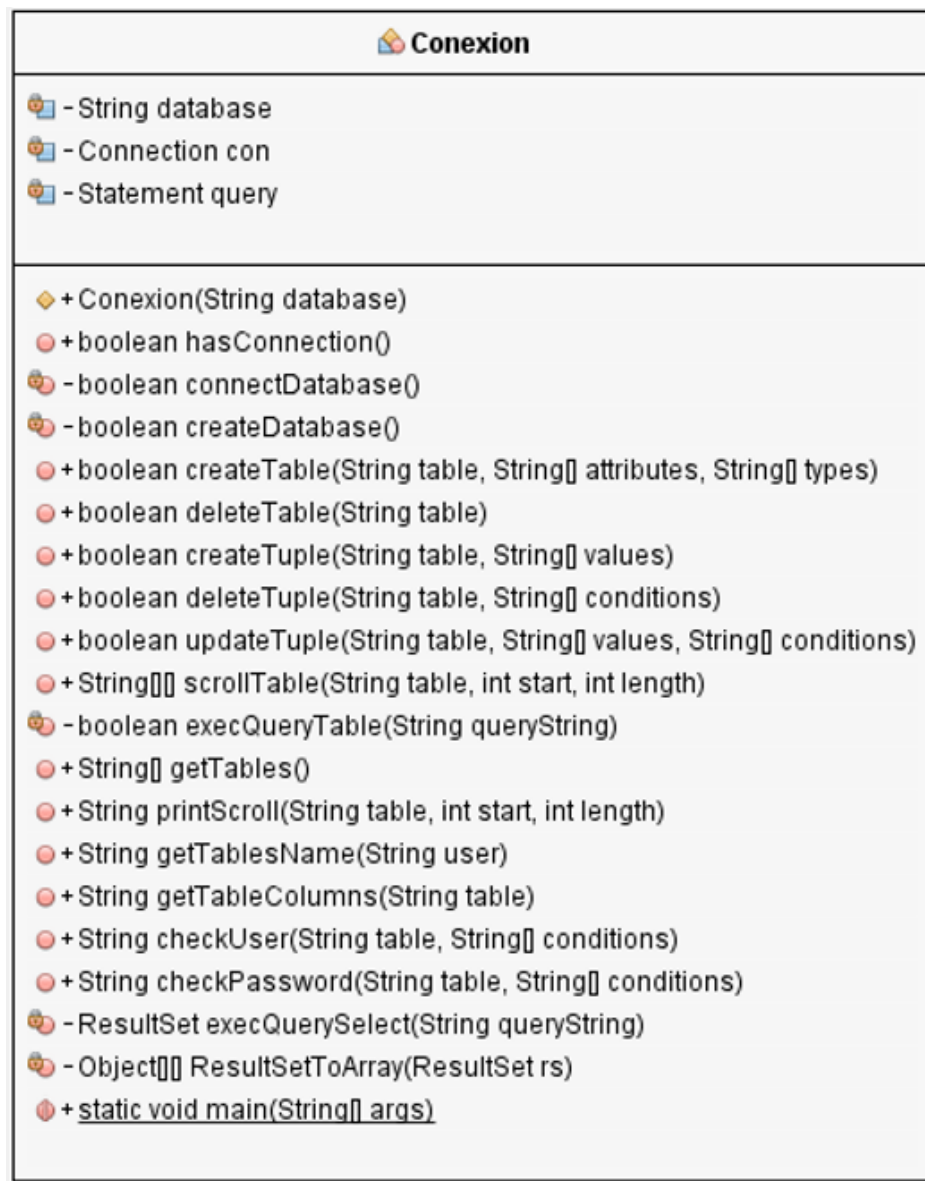


Figura 1: Diagrama UML de clase Conexion

2.1. Diagramas UML de clases



Figura 2: Diagrama UML de clase OperationsResource

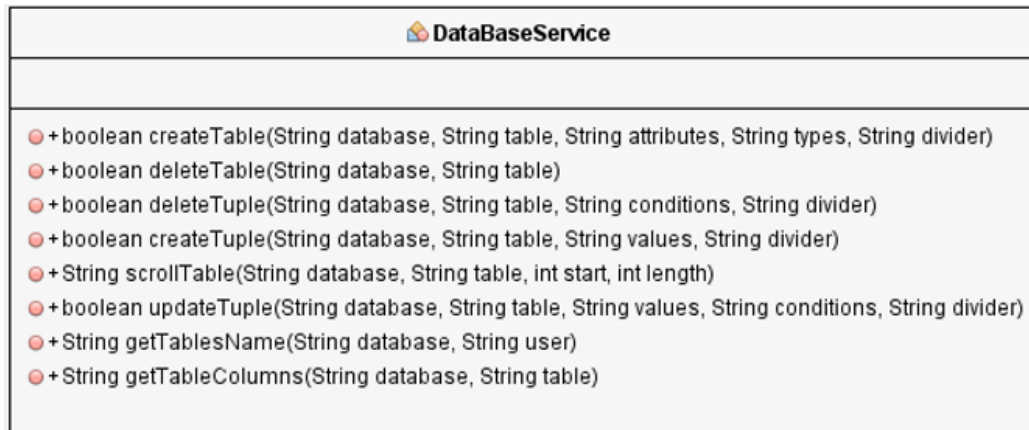


Figura 3: Diagrama UML de clase DataBaseService

2.2. Diagramas jerárquico del proyecto

La figura 4 muestra la organización del sistema "Data Web Wizard". El sistema se implementó utilizando los siguientes tres proyectos web: OmegaClient, OmegaRest y OmegaSoap. En primer lugar, OmegaRest contiene los servicios RESTful mientras que OmegaSoap tiene los servicios SOAP. En segundo lugar, OmegaClient es el proyecto principal que llama a los servicios web RESTful del proyecto OmegaRest que a su vez utiliza los servicios SOAP del proyecto OmegaSoap.

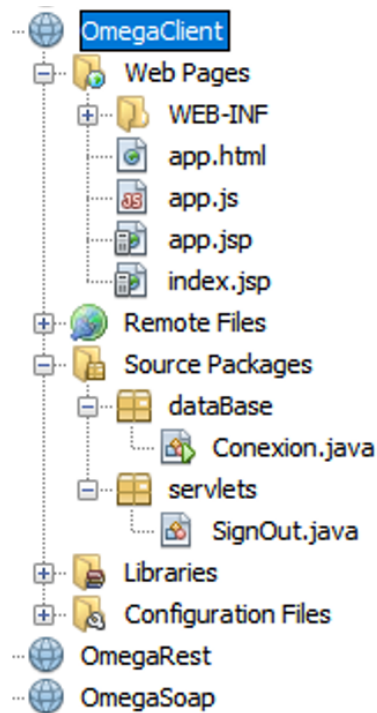


Figura 4: Organización del proyecto

2.3. Diagrama de secuencia

La clase Conexión maneja la base de datos y contiene métodos para agregar, eliminar, consultar y modificar tablas o tuplas. Para el manejo de la base de datos, la clase Conexión es llamada por la clase DataBaseService que contiene servicios web SOAP (Simple Object Access Protocol). Más adelante, los servicios SOAP de DataBaseService son llamados por servicios RESTful de la clase OperationResource.

La figura 5 muestra el diagrama de secuencia del proyecto. En verde se muestra la secuencia para iniciar sesión y en rosa la sesión de las consultas a la base de datos.

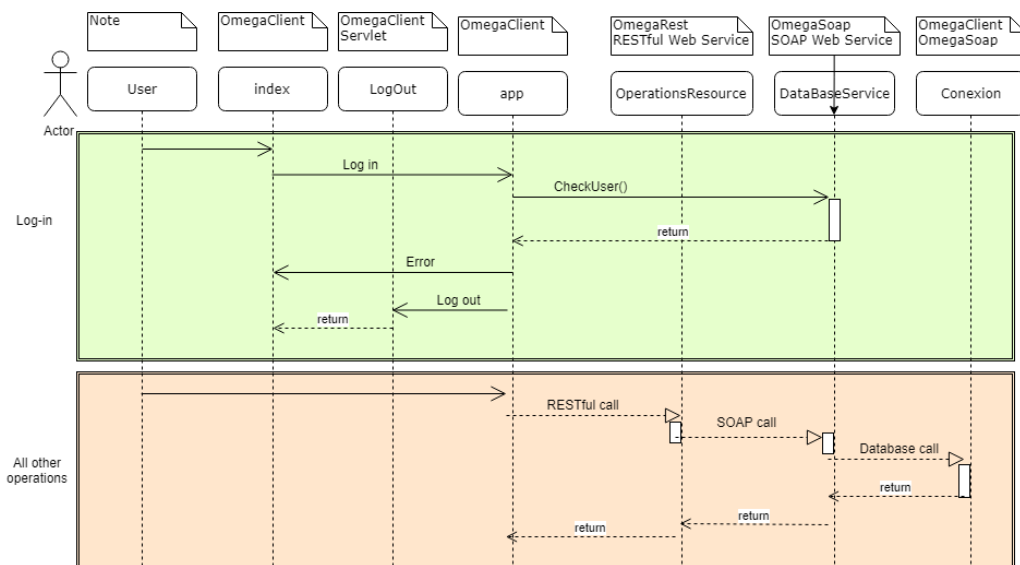


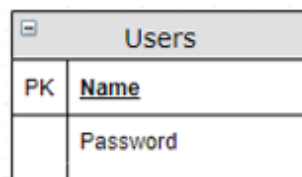
Figura 5: Diagrama de secuencia del proyecto

Para iniciar sesión, el usuario da su nombre y contraseña en la página index y la página app.jsp se encarga de llamar a la clase Conexión para verificar si el usuario se encuentra en la base o no. Si el usuario no estaba previamente registrado, lo registra. En caso contrario, si el usuario ya estaba registrado,

el jsp checa que la contraseña sea correcta. Por un lado, si la contraseña es incorrecta, redirecciona al usuario a la página index con un mensaje de error. Por otro lado, si la contraseña es correcta, crea una sesión HttpSession y carga la página app con los datos y tablas del usuario, permitiéndole modificar o consultar la base de datos.

2.4. Diagrama de bases de datos

La figura 6 muestra el diagrama de la tabla por default en la base de datos. La tabla de “Users” almacena los usuarios existentes y sus contraseñas. Como se ve en la figura 6, la llave primaria es el nombre (name). Por lo tanto, si dos usuarios tienen el mismo nombre, el segundo usuario no podrá registrarse solo con su nombre, deberá registrarse con un alias. Por ejemplo, si un usuario se registra como “Victor” el segundo usuario se deberá registrar como “Victor1” o alguna alteración del nombre “Victor”.



Users	
PK	<u>Name</u>
	Password

Figura 6: Tabla de usuarios en base de datos

3. Descripción de los obstáculos encontrados y cómo fueron resueltos

3.1. Iniciar sesión mediante un Servlet

La solución inicial incluía un servlet de log in intermedio entre la página de inicio (index.jsp) y la página principal (app.jsp). El objetivo del servlet era hacer las consultas a la base de datos utilizando la clase conexión. En caso de que el usuario ingresará sus credencial de forma incorrecta, el servlet era el encargado de redireccionar la solicitud a la página inicial index.jsp. El obstáculo encontrado fue que para iniciar sesión se hacían dos brincos por lo tanto la solución fue eliminar el servlet de inicio de sesión y agregar esa funcionalidad a la página principal.

3.2. Desplegar los servicios web

En un inicio, se refrescaban las páginas .html y .jsp en el navegador para ver los cambios y las actualizaciones. Sin embargo, los cambios en los servicios SOAP no se refrescaban automáticamente en el navegador. La solución fue desplegar (“deployar”) los servicios web manualmente después de cada cambio significativo.

3.3. Actualizar SOAP

Un gran problema fue que los cambios en los servicios SOAP o los nuevos métodos en los servicios no parecían accesibles desde los servicios RESTful en el proyecto OmegaRest y como consecuencia, el cliente, dentro del

proyecto OmegaClient, tampoco podía acceder a los servicios actualizados. Inicialmente, se creaba un nuevo clientes web (New Web Service Client) con cada cambio. Sin embargo, descubrimos el botón de Refresh” de la figura 7 que actualiza los cambios de los servicios SOAP.

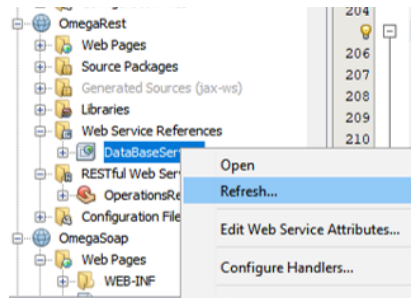


Figura 7: Tabla de usuarios en base de datos

3.4. Parámetros en servicios Web RESTful

Los parámetros en los métodos RESTful se pasaron en una cadena (String) conteniendo un JSON. Dentro de cada método, se utilizó la función JSONParser para recrear el objeto JSON utilizando la cadena recibida. Sin embargo, el método deleteJSON no aceptó la cadena para convertirla en objeto JSON. La solución fue utilizar la directiva `@PathParam("json")` antes de la cadena.

3.5. Cargar el JavaScript

4. Lista de Bugs conocidos

- **Clases de Java de JSON** Se agregó la librería java-json para utilizar métodos de manipulación de archivos JSON. Sin embargo, en ocasiones, al darle click en “clean and build” aparece el error “unable to delete

java-json.jar” dado que parece estar en uso aunque no lo esté. La solución temporal al bug es cerrar NetBeans y borrar el archivo jar en la carpeta “build” del proyecto. Se cree que este error es el resultado de incompatibilidad de versiones en NetBeans ya que en proyecto se creó en la versión 8 y el error aparecía en la versión 8.2.

- **Log out** Existe un servlet que inhabilita la sesión HttpSession, sin embargo, una vez que el usuario ha cerrado sesión y se encuentra en la página inicial para ingresar sesión, si el usuario da click para volver a la página anterior en el navegador, se recupera la sesión.

5. Conclusion

El proyecto “Data Web Wizard” permitió implementar un sistema de manejo de bases de datos dinámicas utilizando servicios web RESTful y servicios web SOAP. Asimismo, el proyecto reforzó el manejo de sesiones utilizando HttpSession y fue una oportunidad para descubrir nuevas herramientas de desarrollo web como Bootstrap.