Eugenio Gonzalo Jiménez

Santiago Cervantes Reus

Eugenio90@gmail.com

yagowolf47@gmail.com

Diseño e implementación de un servicio web RESTful

Práctica Sistemas Orientados a Servicios

# Diseño del servicio

Para el diseño del servicio optamos por la solución más intuitiva: La primera parte del URI identifica el recurso y a continuación se usaría el identificador del usuario en caso de hacer falta, de forma que para hacer referencia a un usuario, el URI a utilizar sería: usuarios/{UserID}. De la misma forma, el URI a utilizar para el caso de hacer referencia a un calendario sería: calendarios/{UserID}/{CalID}. Una cita se identificaría de la misma manera solo que, obviamente, cambiando los nombres.

De esta forma, cada objeto se encuentra identificado de forma única pero, al mismo tiempo, de una forma muy simple y sencilla de manejar.

Consideramos que el diseño de las URI’s de un servicio puede llegar a ser bastante complejo, obliga a diseñar por anticipado una cierta arquitectura y mantener en todo momento un nivel de abstracción muy completo sobre el servicio que se está construyendo; En este caso, el ejemplo no era demasiado complejo por lo que consideramos que nuestra solución con respecto al diseño de las URI’s es la más común y, por qué no, obvia.

A continuación, presentaremos las tablas del servicio desarrollado:

|  |  |
| --- | --- |
| **URI** | **http://localhost:8080/Restfull/webresources/usuarios** |
| **Método** | GET |
| **Devuelve** | Lista de usuarios. List<Usuarios> XML Schema |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/ webresources/usuarios/{UserID}** | |
| **Método** | GET | |
| **Devuelve** | Usuarios, XML Schema | |
| 404 | Usuario no encontrado |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/ webresources/usuarios/{UserID}** | |
| **Método** | DELETE | |
| **Devuelve** | 200 | Ok |
| 404 | Usuario no encontrado |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/ webresources/usuarios** | |
| **Método** | POST | |
| **Cuerpo** | XML Schema del usuario a crear | |
| **Devuelve** | 201 | Created + URI del objeto creado |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/webresources/calendarios** | |
| **Método** | POST | |
| **Cuerpo** | XML Schema del calendario a crear | |
| **Devuelve** | 201 | Created + URI del objeto creado |
| 409 | Conflict + URI del objeto motivo del conflicto |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/calendarios/{UserID}/{CalID}** | |
| **Método** | GET | |
| **Devuelve** | Calendario: XML Schema del calendario. | |
| 404 | Usuario no encontrado |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/calendarios/{UserID}/** | |
| **Método** | GET | |
| **Devuelve** | List<Calendario>: XML Schema de los calendarios. | |
| 404 | Usuario no encontrado |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/calendarios/{UserID}/{CalID}** | |
| **Método** | PUT | |
| **Cuerpo** | XML Schema del calendario a modificar | |
| **Devuelve** | 202 | Accepted |
| 404 | Usuario no encontrado |
| 404 | Calendario no encontrado |
| 409 | Calendario no pertenece al usuario indicado |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/calendarios/{UserID}/{CalID}** | |
| **Método** | DELETE | |
| **Devuelve** | 200 | Ok |
| 404 | Usuario no encontrado |
| 404 | Calendario no encontrado |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/webresources/citas** | |
| **Método** | POST | |
| **Cuerpo** | XML Schema de la cita a crear | |
| **Devuelve** | 201 | Created + URI del objeto creado |
| 409 | Conflict + URI del objeto motivo del conflicto |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/citas/{IDUsuario}/{IDCita}** | |
| **Método** | PUT | |
| **Cuerpo** | XML Schema de la cita a modificar | |
| **Devuelve** | 200 | Ok |
| 404 | Cita no encontrada |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/citas/{IDUsuario}** | |
| **Método** | GET | |
| **Cadena de consulta** | Hasta= | Fecha hasta la cual se desea filtrar las citas del usuario, formato (yyyy-MM-dd) |
|  | Desde= | Fecha a partir de la cual se desea filtrar las citas del usuario, formato (yyyy-MM-dd) |
|  | IDCalendario= | ID del calendario |
|  | maxEntradas= | Máximo número de entradas resultado del filtrado |
| **Devuelve** | List<Citas>, XML Schema de la lista | |
| 404 | Usuario no existe no encontrada |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/citas/{IDUsuario}/{IDCalendario}** | |
| **Método** | GET | |
| **Devuelve** | List<Citas>, XML Schema de la lista | |
| 404 | El calendario no pertenece al usuario indicado |

|  |  |  |
| --- | --- | --- |
| **URI** | **http://localhost:8080/Restfull/citas/{IDUsuario}/{IDCita}** | |
| **Método** | DELETE | |
| **Devuelve** | 200 | Ok |
| 404 | Cita no encontrada |

# Diseño de la BBDD

Con respecto a la base de datos, esta ha sido construida mediante la herramienta incluida en NetBeans JavaDB. Que dicha herramienta estuviese incluida dentro del propio IDE ha resultado ser una gran ayuda ya que nos ha permitido controlar de forma más directa y completa lo que ocurría con los datos al usar el servicio o ejecutar el cliente.

IDcalendario : integer

Descripcion : varchar(50)

Fecha : Date

ID : integer

**Calendarios**

Propietario : integer

Nombre : varchar(50)

**Usuarios**

ID : integer

Pass : integer

Nombre : varchar(50)

Publico : boolean

ID : integer

**Calendarios**

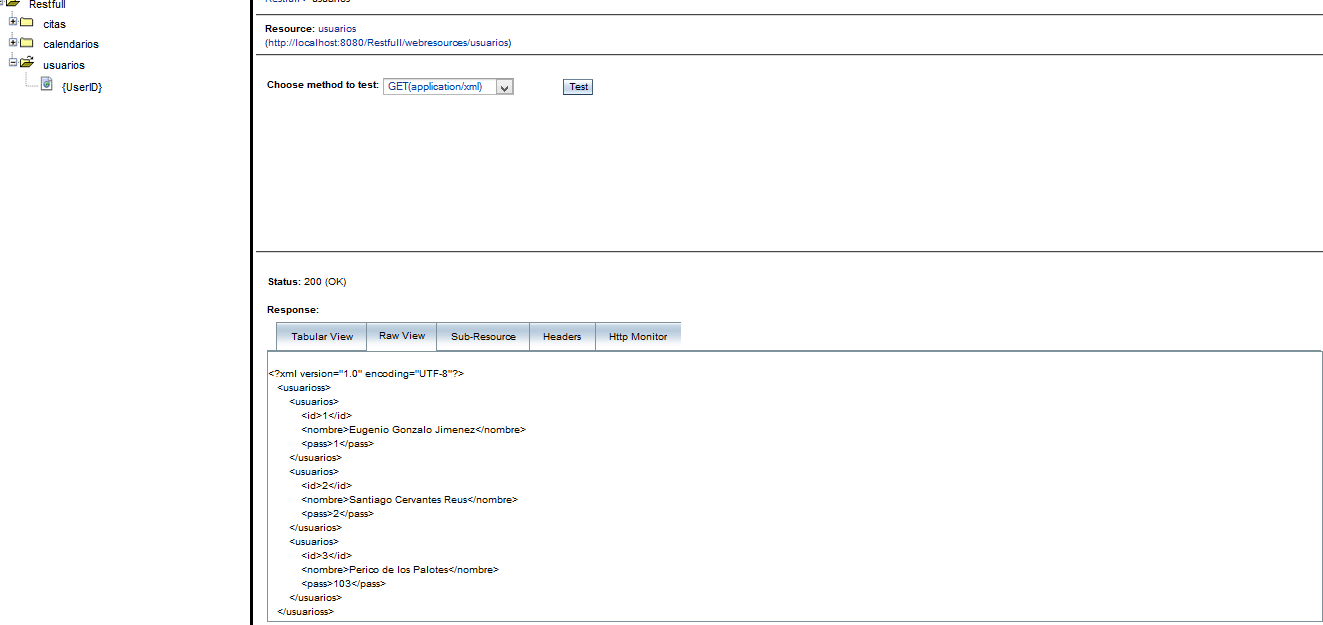
Como se puede observar, la base de datos está dividida en tres tablas cada una encargándose de un propósito distinto además de mantener una relación entre las distintas tablas más que asequible.

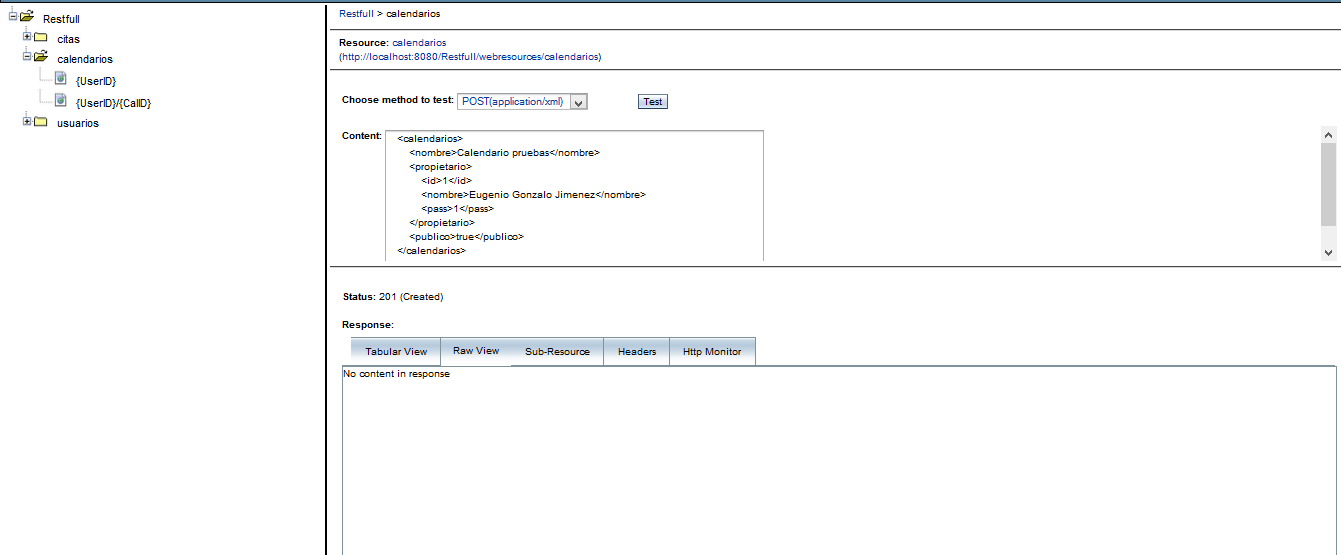
Los ID de cada tabla (la clave) se generan automáticamente, asignándoles el último número utilizado incrementado en uno.

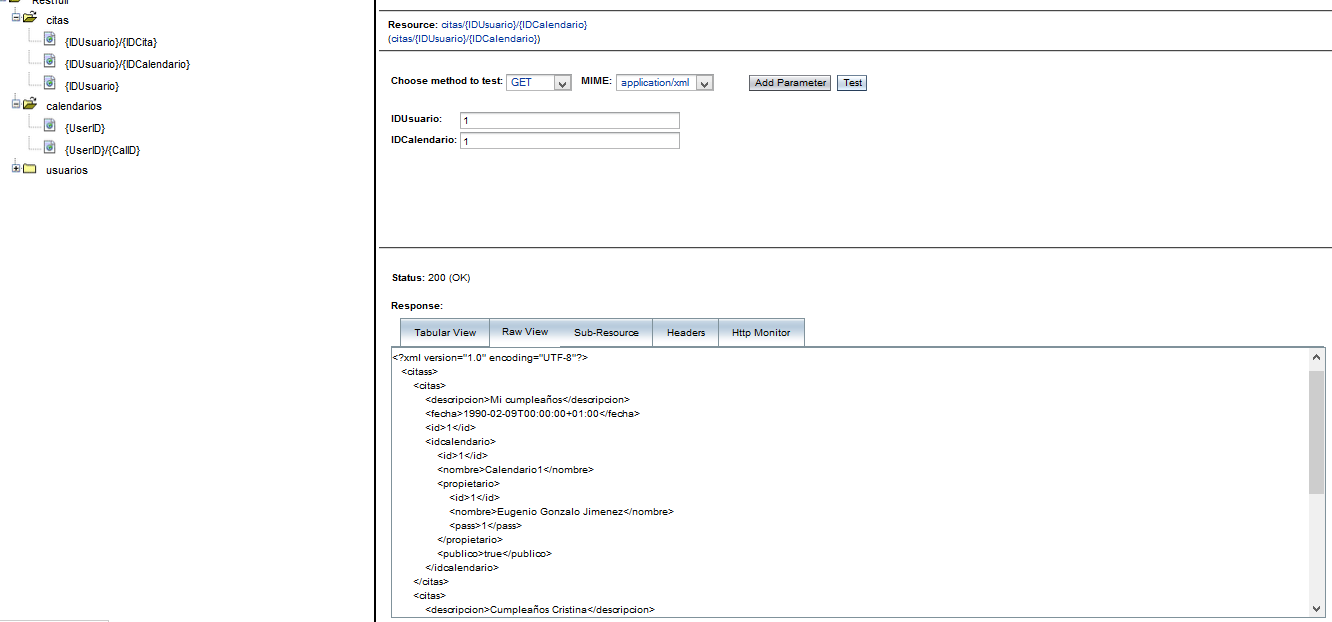
Un detalle es el uso del tipo Date de sql, que nos permite guardar las fechas como fechas y facilita el interactuar con la base de datos.

# Capturas de Rest Client

Por sencillez, no se realizarán capturas de todos y cada uno de los métodos si no de solo unos pocos que sirvan como resumen del servicio. Esto no es porque aquellos cuya captura no aparezca no funcionen, si no por no extender este documento más allá de unos límites razonables.







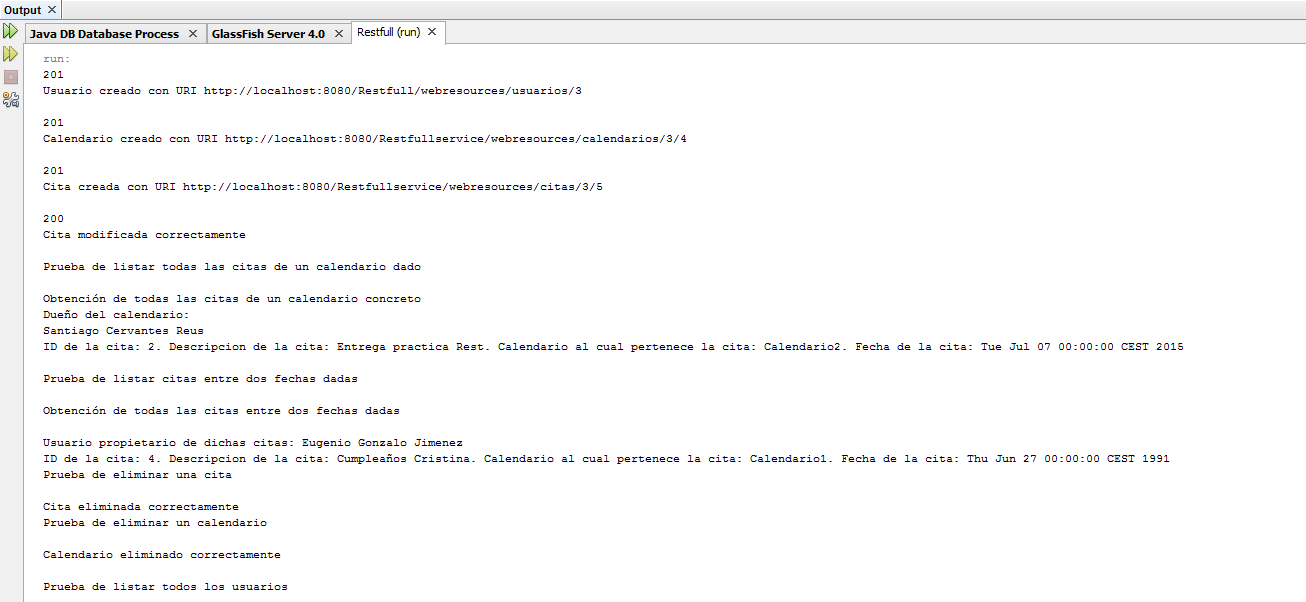
En esta última captura se puede observar como aún hay más citas, pero las limitaciones de la pantalla me impiden mostrarlo.

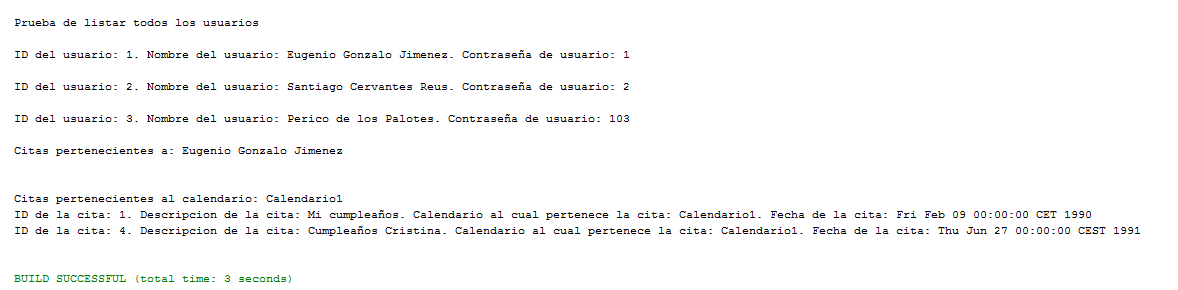
Hemos preferido evitar realizar una captura de una operación de modificación (PUT) ya que entre los campos y la respuesta serían aún más complicado que se mostrase.

# Cliente de prueba

Se pegan la captura de los resultados obtenidos por el cliente de pruebas, utilizado para comprobar el funcionamiento, en código Java, del servicio desarrollado. Por sencillez el cliente está hecho de forma artificial para probar ciertos IDs o ciertos parámetros muy concretos: Esto ha sido así para comprobar de forma más flexible los casos de acierto y error (Solo necesitábamos cambiar un número) pero, en cualquier caso, podría utilizarse el propio servicio para obtener los IDs de los objetos o su URI para posteriormente acceder a ellos; Es decir, el que el cliente esté desarrollado como un conjunto de pruebas estricto no supone un menoscabo del funcionamiento del servicio, solamente una peculiaridad en su desarrollo.

La única función del cliente ha sido comprobar de forma externa el funcionamiento del servicio por lo que, en cualquier caso, cumple con su cometido.





Las dos capturas anteriores están hechas de forma que se solapen, para poder mostrar todas las pruebas que nos eran requeridas.

# Comentarios personales

Consideramos que, aunque servían como un punto de inicio aceptable, la información proporcionada sobre cómo trabajar con Rest o sobre cómo enfrentar ciertos problemas propios del traspaso de información ha sido insuficiente: Ha sido más labor del equipo el descubrir cómo resolver los diferentes escoyos que nos surgían. Por otra parte, esta ausencia de información nos ha obligado a investigar y desarrollar un trabajo por nosotros mismos, lo cual nos ha dado una visión algo más profunda y/o general que la que nos hubiera dado la realización de esta práctica de forma guiada.

No puedo dejar de hacer notar un error en la práctica y es que, al modificar los valores de una entrada de la BBDD, en vez de modificarse los valores se introduce una nueva entrada en la BBDD cuyos campos son el resultado de realizar las modificaciones al original pero conserva la versión anterior. Esto solo ocurre al utilizar el cliente de pruebas y no así al usar Rest Client por lo que hemos sido incapaces de resolver dicho error.

Le enviamos un correo al profesor y estuvimos investigando todo lo que pudimos pero no conseguimos encontrar el fallo, por lo que preferimos no bloquear el trabajo debido a un problema de este tipo y realizar el resto de la práctica pedida; El código está correcto, por lo que arreglando eso el servicio pasaría a ser tan válido como el que más.