**Reto 3 Tópicos Especiales en Telemática**

Sebastián García Valencia

Estudiante Ingeniería de Sistemas

Universidad EAFIT

Medellín, Colombia, Suramérica

**Resumen**

El presente documento contiene el análisis y diseño de una calculadora distribuida que usa servicios web tanto REST como SOAP.

**Palabras clave.**

Servicio web, REST, SOAP, glassfish, java

1. **Introducción**

En la actualidad los servicios web son una herramienta ampliamente difundida en los sistemas distribuidos, la pregunta es cuando usar REST y cuando SOAP, con el presente programa se pretende usar ambos para conocerlos un poco mejor y tener criterios a la hora de una elección, es así que esta calculadora distribuida usa SOAP para las operaciones de sumar y multiplicar y REST para restar y dividir.

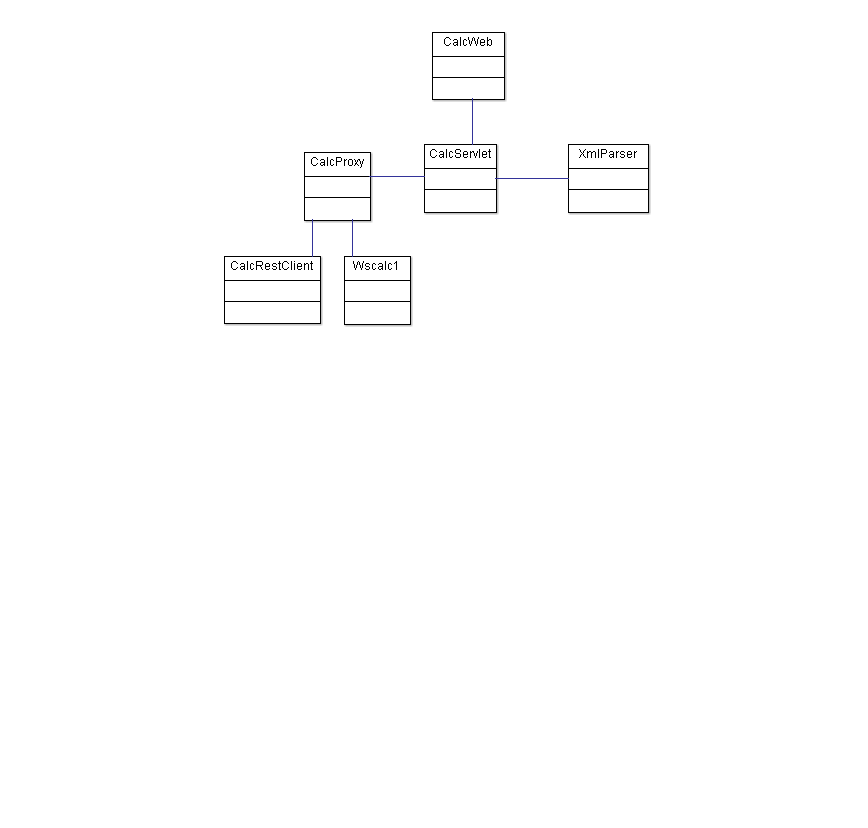
1. **Arquitectura**
   1. **Diseño global**

Como podemos observar se pretende simular 5 nodos distintos, aunque la simulación se hará local, los archivos de configuración permiten llevar el programa a un ambiente distribuido fácilmente.

****

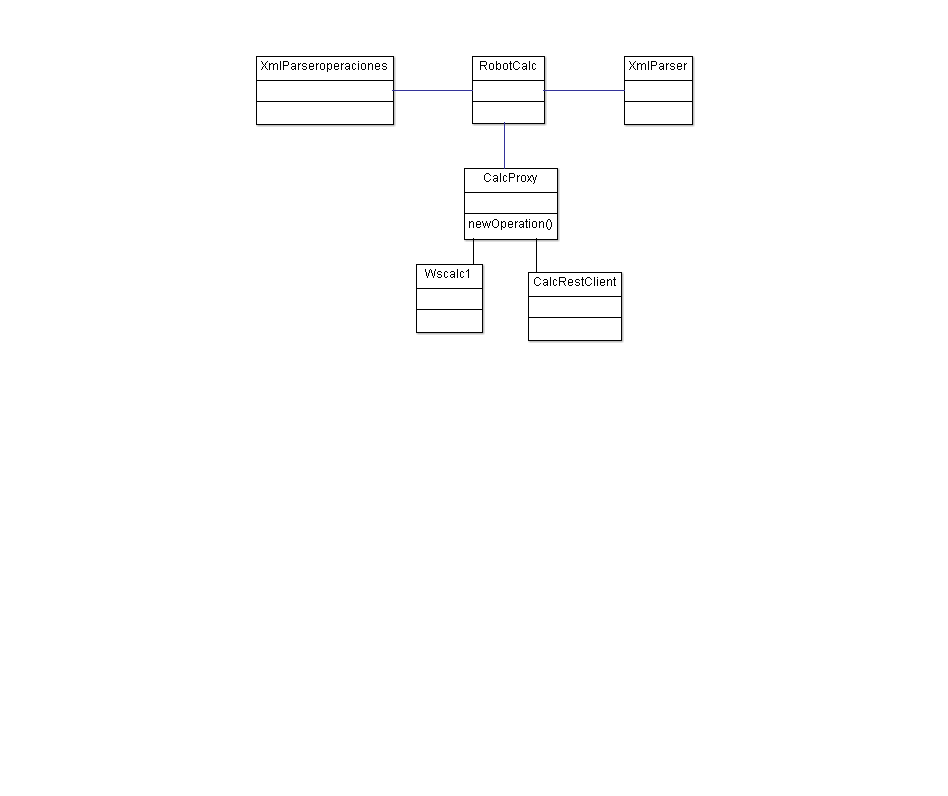
* 1. **Vista lógica CalcWeb**

Esta es la aplicación web que accede a los servicios, como vemos se vale de un servlet para parsear de un xml las urls de los servicios web y por el proxy donde están los clientes para cada servicio se comunica.

****

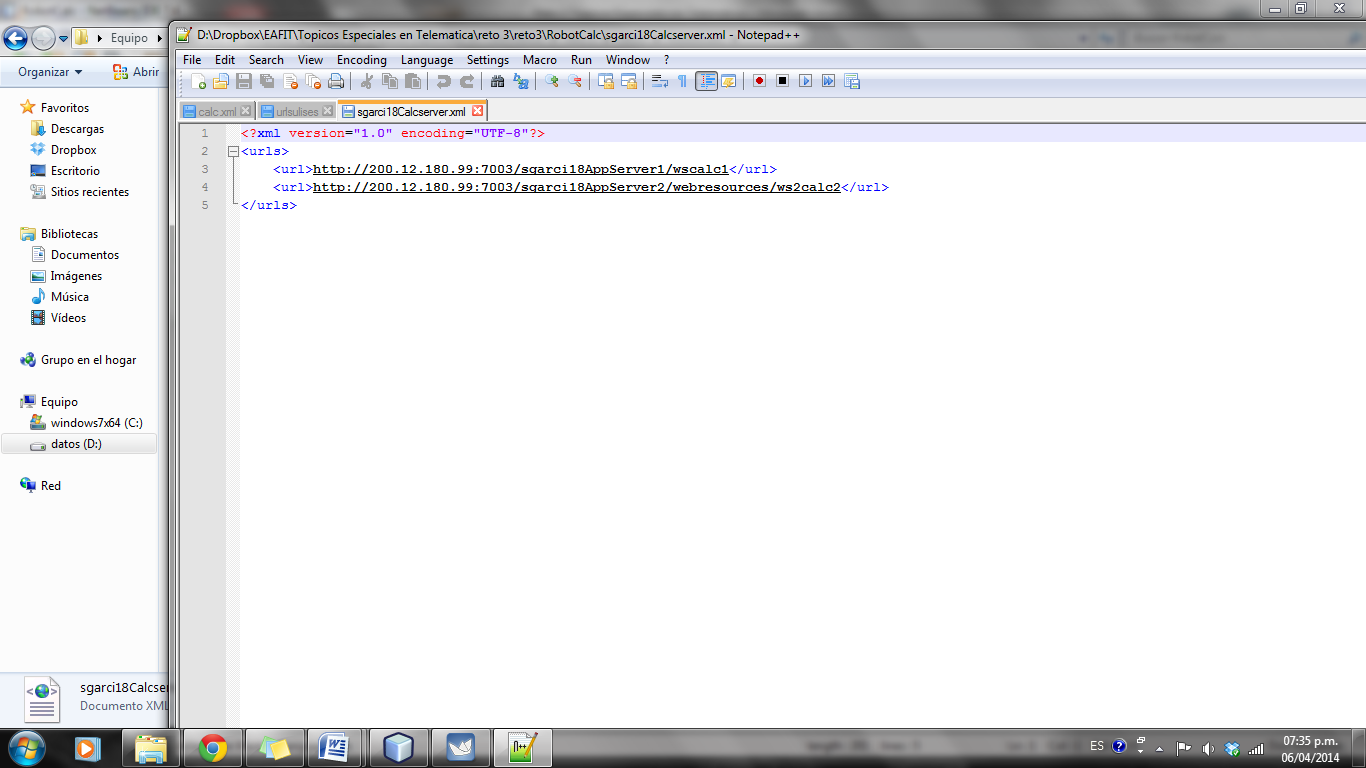
* 1. **Vista lógica RobotCalc**

Hasta cierto punto el robot conserva la mayoría de la arquitectura de la app web, la diferencia es que en este caso la entrada y la salida es un xml por lo cual existe otro parser xml encargado de modificar el archivo como debe ser.

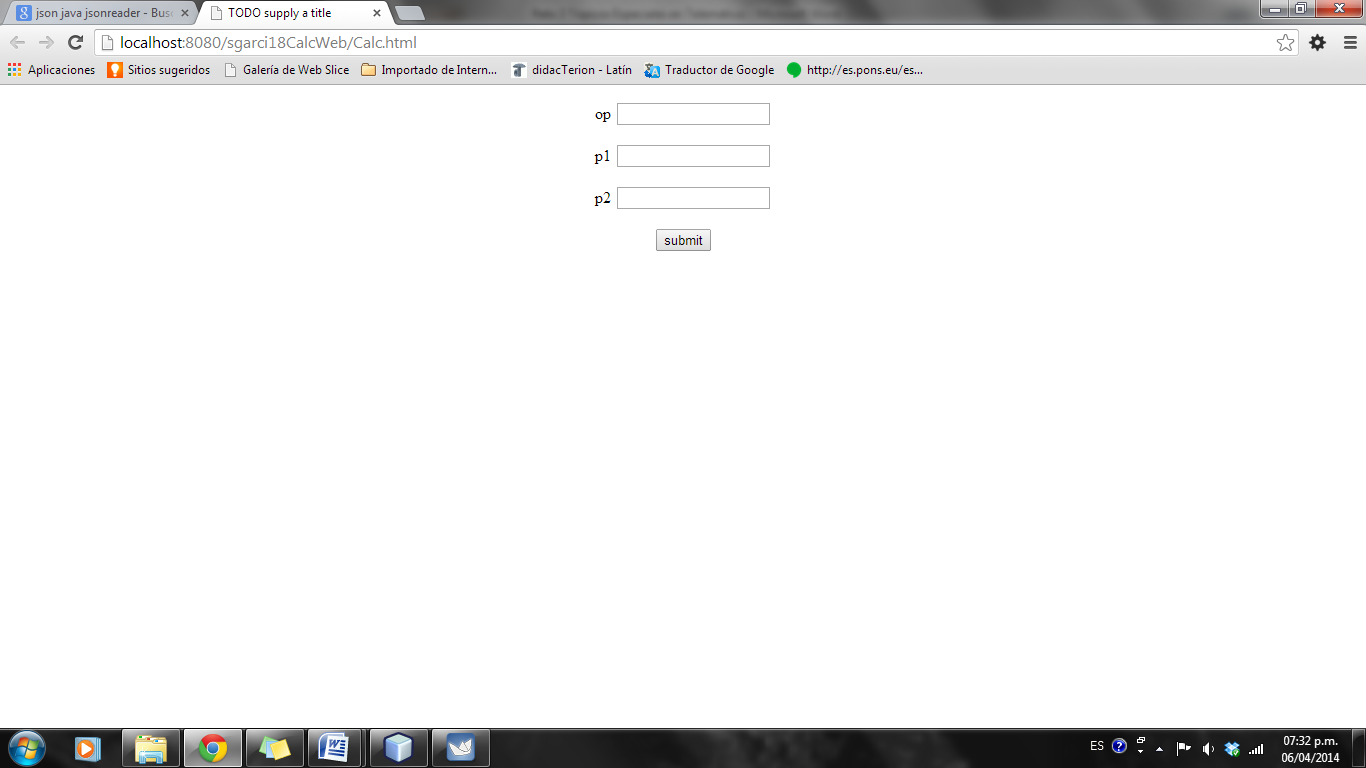
****

1. **Funcionamiento**
   1. **CalcWeb**

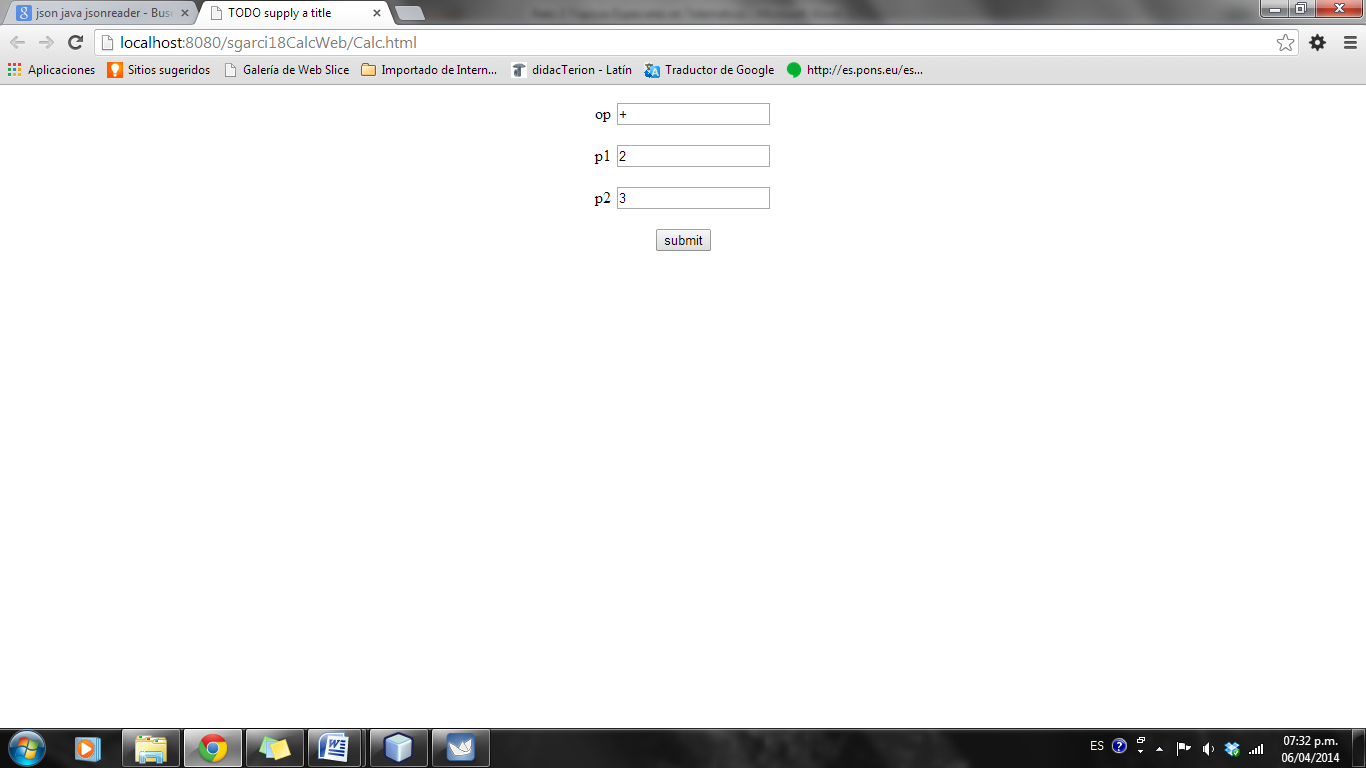
Este el archivo de configuración donde se deben poner las urls de los servicios web (aquí vemos los que están desplegados en el servidor de eafit)

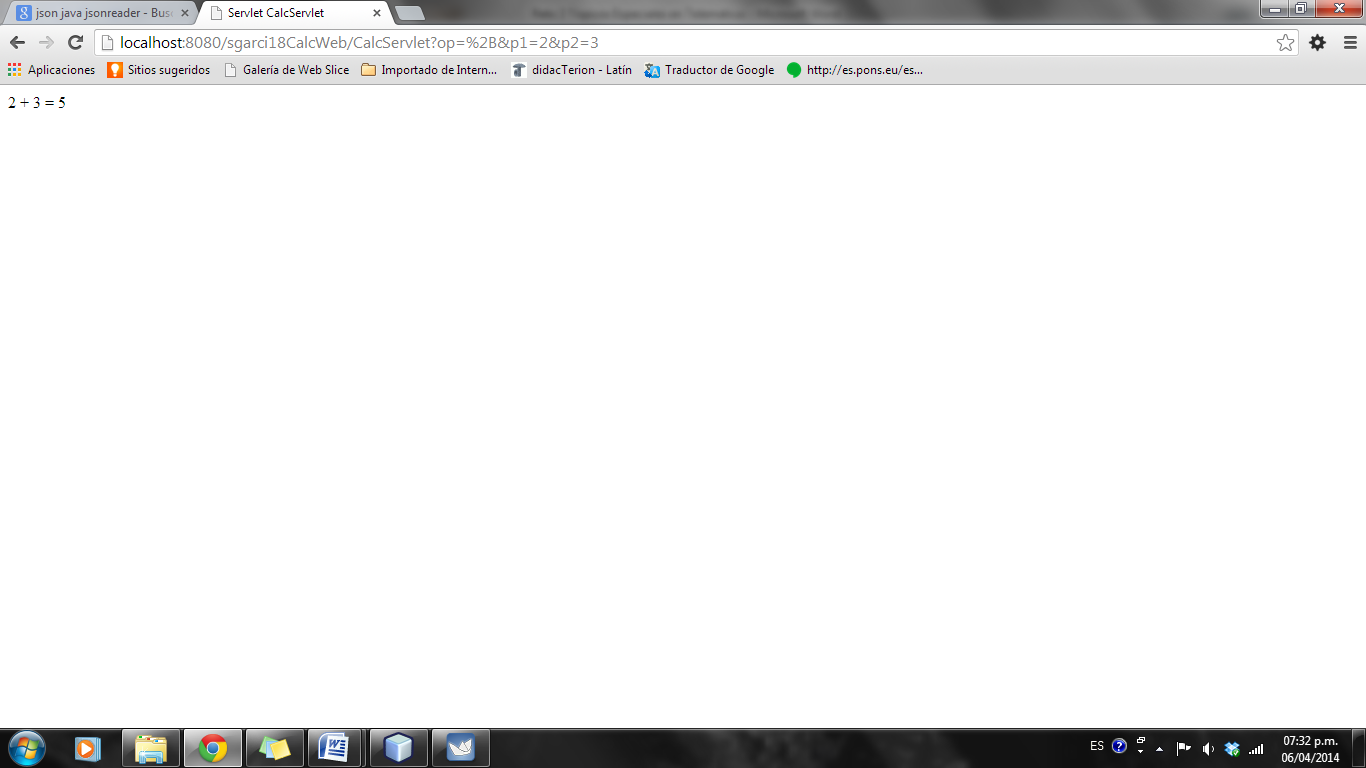
****

La ventana inicial de la aplicación

****

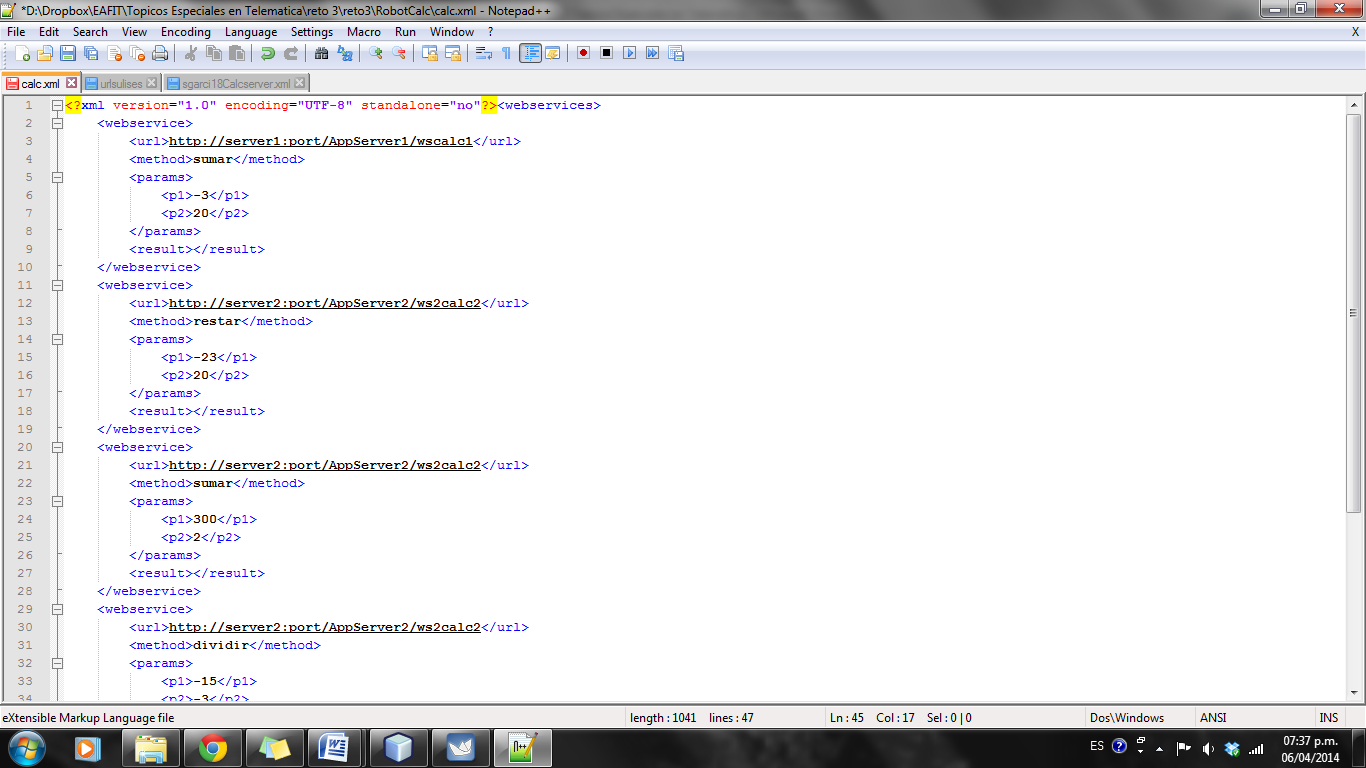
En este caso haremos una suma

****

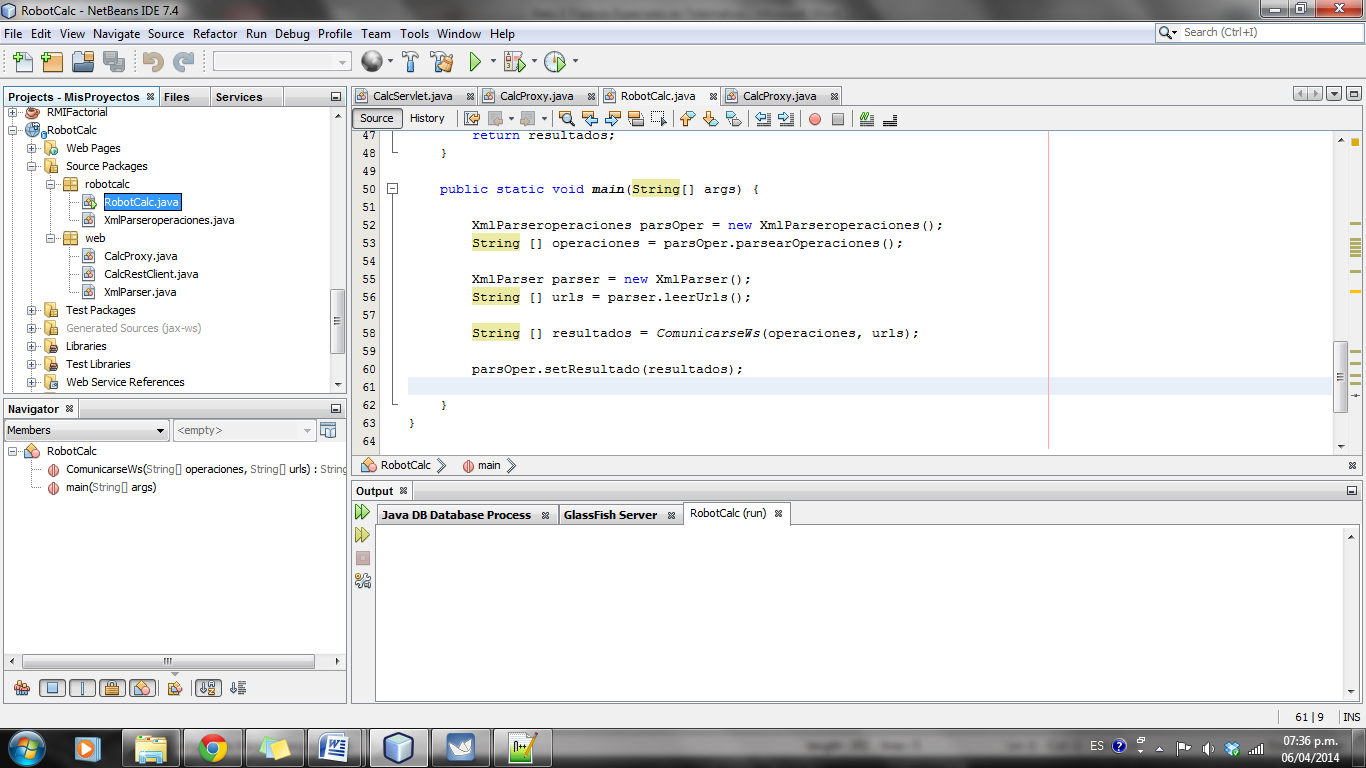
Por debajo se llamo el ws soap que suma, si hubiésemos dividido o restado se habría usado el rest, sin embargo eso es transparente para la aplicación, esta simplemente llama un servlet con los parámetros (como se ve en la url), y finalmente el proxy es el que diferencia y se ensucia las manos. ****

* 1. **RobotCalc**

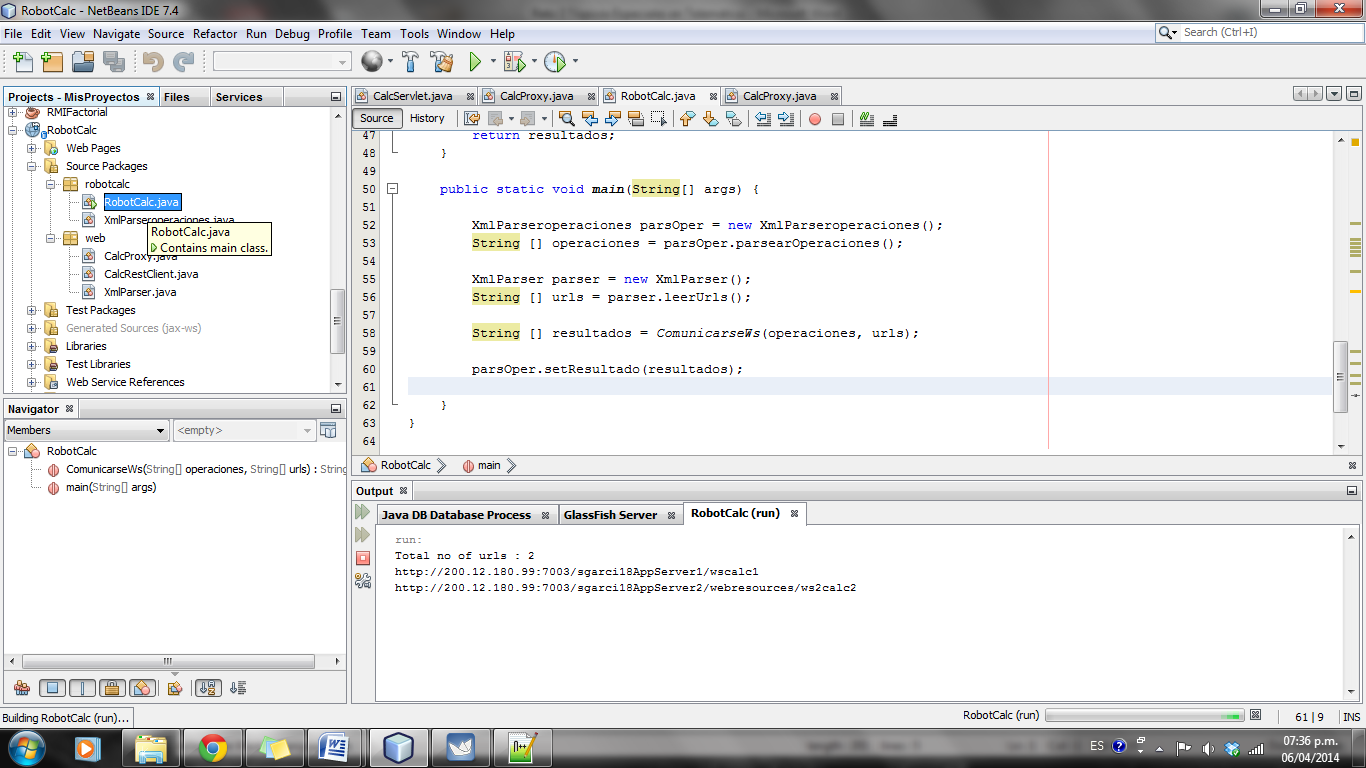
Aquí sucede lo mismo con el xml de las urls, pero adicionalmente está este otro donde están las operaciones y parámetros

****

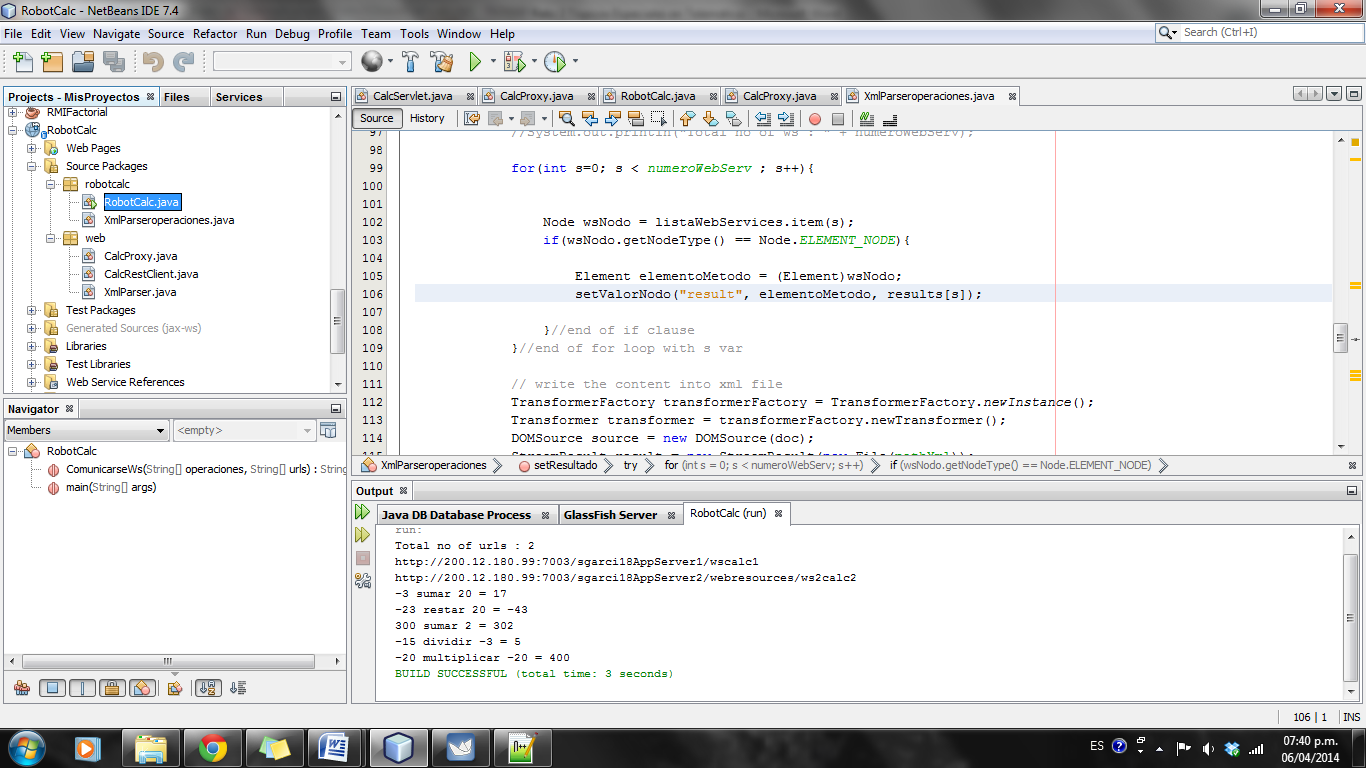
Debemos ejecutar la aplicación RobotCalc que es donde está el main

****

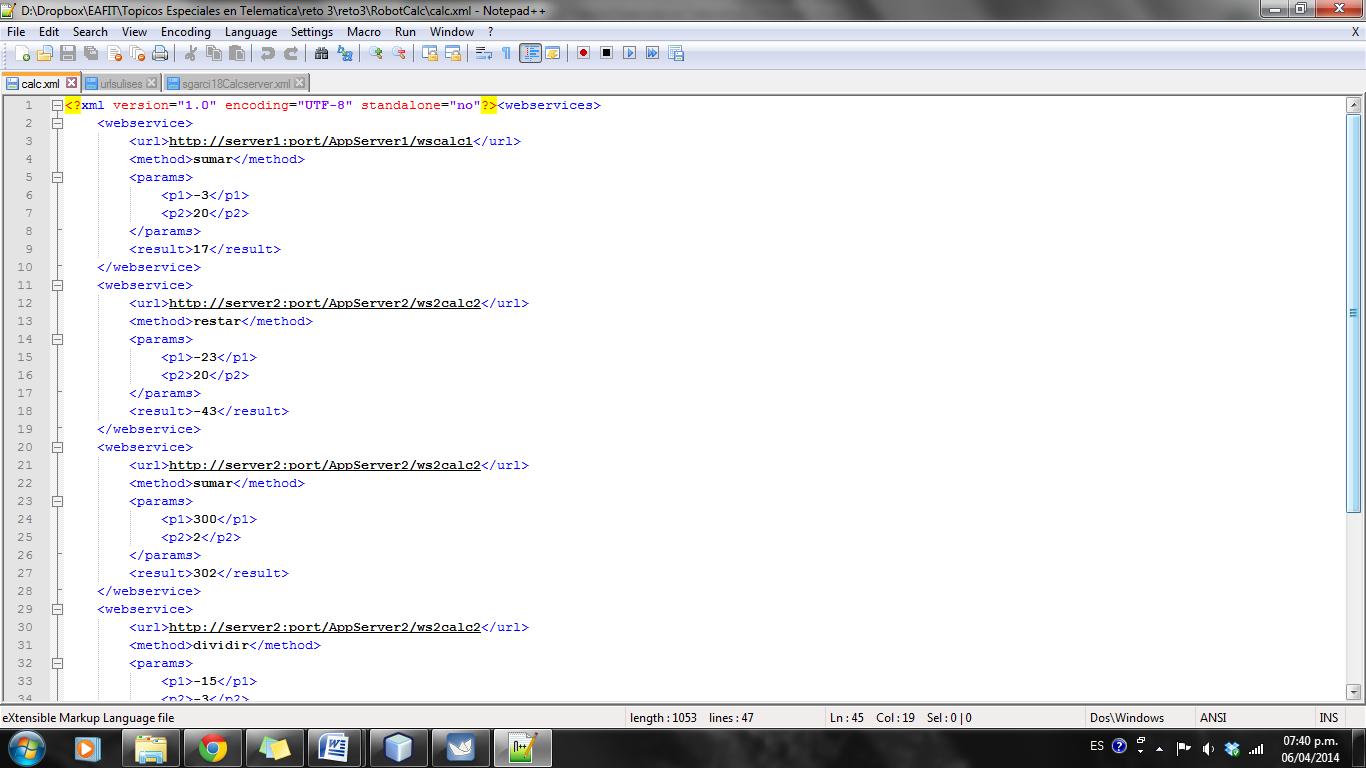
Aquí vemos que antes que nada parsea las urls y establece conexión



Luego usa el respectivo ws para cada operación

****

El resultado final se pone en el xml original

****

1. **Posibles mejoras.**

* A la hora del despliegue, el archivo de configuración donde están las url de los ws tiene el problema de que glassfish no conserva la estructura de archivos original, así que se debe ya sea quemar en código la ruta absoluta, o poner el archivo en la carpeta domain1/config de glassfish, para ahorrarse este trabajo se podrían manejar paths relativos.

1. **Enlaces**

Repositorio GIT: <https://bitbucket.org/sebasgverde/reto3toptelematica>

Página principal CalcWeb: <http://200.12.180.99:7003/sgarci18CalcWeb/Calc.html>

Servicio web SOAP: <http://200.12.180.99:7003/sgarci18AppServer1/wscalc1?wsdl>

Servicio web REST: [http://200.12.180.99:7003/sgarci18AppServer2/webresources/ws2calc2?op=\*&p1=2&p2=2](http://200.12.180.99:7003/sgarci18AppServer2/webresources/ws2calc2?op=*&p1=2&p2=2)

1. **Bibliografía.**

<http://en.wikipedia.org/wiki/Representational_state_transfer>

<http://javarevisited.blogspot.com/2011/12/parse-xml-file-in-java-example-tutorial.html>

<http://www.journaldev.com/2315/java-json-processing-api-example-tutorial>