UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Implementación de un servicio web basado en REST para la medición de la calidad de datos

Proyecto Final de Carrera

Ingeniería Informática

Autor: Pablo Miñarro Martinez

Director: Juan Miguel García Gómez

Co-Director: Carlos Sáez Silvestre

Julio-2014

Resumen

La falta de garantías en la calidad de los datos en el ámbito biomédico puede llevar a tomar malas decisiones en el tratamiento de los pacientes y conclusiones erróneas en los estudios poblacionales. Para contribuir a resolver este problema, presentamos el desarrollo de un servicio web implementado en Java para medir la calidad de los datos biomédicos, conjuntamente con una interfaz accesible desde cualquier navegador web para poder hacer uso del servicio.

Para la medición de la calidad, se ha empleado una aproximación resultante de combinar nueve dimensiones de calidad (Completeness, Consistency, Duplication, Correctness, Temporal Stability, Spatial Stability, y Contextualization), aplicadas a distintos ejes de los datos (Dataset, Attribute, Registry, Value, Source y Time). La arquitectura está diseñada según el patrón estrategia, que permite la implementación del método de medida de cada dimensión sobre la clase abstracta "Axis", que se especializa en los seis posibles ejes. Como prueba de concepto, se ha procedido a la implementación de una dimensión de conteo llamada completitud, y otra basada en reglas, llamada consistencia.

Las tecnologías utilizadas para poder realizar el proyecto han sido principalmente el entorno de desarrollo integrado Eclipse, el sistema de gestión de reglas de negocio Drools, el conjunto de herramientas para la implementación de servicios web Jersey, y el framework para la creación de interfaces gráficas de usuario web Vaadin.

En el desarrollo se han respetado las condiciones de escalabilidad y rendimiento, que requiere un proyecto destinado al manejo de un volumen de datos que puede llegar a ser elevado. Además, se ha logrado realizar todo el proyecto empleando bibliotecas de código libre.

Palabras clave: data, quality, biomedical, webservice , drools, vaadin, jersey

Tabla de contenidos

1. Introducción

En la llamada sociedad de la información, los datos tienen un papel protagonista debido a que es a partir de donde se extrae conocimiento. Dicha información no para de crecer en cantidad cada día gracias a los avances en tecnología, que permiten controlar nuevos parámetros que previamente no eran registrados, y también al aumento del uso del llamado BigData, término que define grandes y complejas colecciones de datos difíciles de manejar, para fines tanto comerciales como industriales, usado por un amplio abanico de sectores. Es dentro del sector o ámbito biomédico donde los datos adquieren una relevancia vital, debido al carácter de su procedencia y al uso al que están destinados. Es por este motivo por lo que la calidad de estos datos debe ser máxima y estar garantizada, para poder reducir las pérdidas en un sector donde no sólo son de carácter económico, y donde el uso de herramientas como la que se presenta puede ayudar a conseguirlo.

1.1. Ámbito

Actualmente, los sistemas de salud disponen de abundante información clínica referente a los pacientes y los estudios realizados, para el uso primario de esta información. Las historias clínicas de los pacientes en formato electrónico (EHR, electronic health records) pueden ser usados para crear repositorios de datos, utilizados para extraer conocimiento gracias a toda la información almacenada para usos secundarios. Estos repositorios deben de ofrecer una garantía de calidad en los datos, ya que la falta de garantías en la calidad de estos datos en el ámbito biomédico, es un problema que puede afectar tanto a las practicas clínicas como a las investigaciones que reutilizan estos datos, debido a que los errores en los datos pueden llevar a errores en el diagnóstico o a conclusiones incorrectas. Por lo tanto, se considera necesario la definición de un framework que garantice cierto grado de calidad de datos, y la creación de herramientas que faciliten la aplicación de este framework. En esta memoria se presenta la plataforma desarrollada para medir la calidad de datos biomédicos, desarrollada como una plataforma con arquitectura orientada al servicio [2] [5] [6] (Service-Oriented architecture, SOA).

Se han desarrollado muchos estudios en el dominio biomédico sobre los métodos para la evaluación de la calidad de datos [1]. Estos métodos están basados principalmente en la medida de dimensiones de calidad, sin embargo, no hay un consenso general sobre qué realizar las medias, y especialmente como hacerlo. En este proyecto se seguirá la aproximación realizada por un estudio en desarrollo [3] [4] dentro de la UPV, en el grupo de investigación de informática biomédica (IBIME), perteneciente al instituto ITACA, que identifica un conjunto de nueve dimensiones donde se puede medir la calidad. Estas dimensiones son, en inglés, completeness, consistency, duplication, correctness, temporal stability, spatial stability, contextualization, predictive value y reliability y en castellano, completitud, consistencia, duplicidad, correctitud, estabilidad temporal, estabilidad espacial, contextualización, valor predictivo y fiabilidad, respectivamente. Además, cada dimensión se puede aplicar a distintos ejes de datos, que tienen de nombre: registry,

attribute, value, dataset, time y source, llamadas en castellano registro, atributo, valor, dataset, tiempo y fuente.

1.2. Objetivo del proyecto

El objetivo principal del proyecto es la creación de un servicio web de tipo REST, para la medición de la calidad de los datos en el ámbito biomédico, accesible tanto por una interfaz web creada para tal efecto, como por terceras aplicaciones. Un objetivo secundario del proyecto es además, comprobar la factibilidad de implementación de dimensiones basadas en conteos y en reglas semánticas, motivo por el cual se deberán implementar totalmente las dimensiones conocidas como "completitud" y "consistencia". Además, el desarrollo deberá implementarse como un punto de partida para futuras y nuevas ampliaciones, por lo que será necesario tener en cuenta un rendimiento lineal, (para conjuntos de datos de tipo medio de 10.000 registros) y prepararlo para lograr un alto grado de escalabilidad, utilizando preferiblemente para la creación del proyecto patrones adecuados de diseño y tecnologías con licencias de código libre.

1.3. Contribuciones

El proyecto se ha realizado en colaboración con el grupo de informática biomédica (IBIME) perteneciente al ITACA, durante el periodo de octubre de 2013 a julio de 2014. Durante la realización del proyecto, se ha logrado publicar los avances realizados hasta principios de 2014 en la contribución "Towards a service oriented platform for the customized analysis of biomedical data quality", Pablo Miñarro, Carlos Saez, Juan Miguel García-Gómez, publicado en el congreso WIICT-2014.

Se ha desplegado la herramienta desarrollada en un servidor de pruebas perteneciente al grupo IBIME, desde donde se puede comprobar sus funcionalidades[1] y acceder al servicio web[2].

2. Calidad de los datos

Para medir la calidad de los datos, utilizaremos el framework propuesto, que emplea distintas dimensiones de calidad [3] y ejes donde aplicar estas dimensiones dependiendo del nivel al que queramos medir los datos. Las dimensiones y ejes se detallan en las siguientes secciones.

[1] Servicio accesible desde en la dirección: http://158.42.166.230:8080/DQV_GUI/

[2] El estado del WebService se pude realizar desde: http://158.42.166.230:8080/dqv_service/rest/test

2.1. Dimensiones

La aplicación de dimensiones para medir la calidad de los datos permite caracterizar los posibles fallos de calidad que pueden estar presentes en los datos analizados, la separación en nueve dimensiones distintas permite analizar y detectar individualmente errores en los datos, dependiendo de las características de dichas dimensiones. Las distintas dimensiones existentes en el framework son:

- Completitud (Completeness): Es el grado en el que datos relevantes se encuentran registrados, y de esta manera se detectan los valores que no se encuentran presentes.

- Consistencia (Consistency): Es el nivel en que los datos cumplen con requisitos y reglas específicas.

- Duplicación (Duplication): El grado en el que los datos contienen registros duplicados representando a la misma entidad.

- Corrección (Correctness): El grado de precisión del dato representado respecto al estado en el mundo real.

- Estabilidad temporal (Temporal stability): El grado de estabilidad de los datos respecto al tiempo.

- Estabilidad espacial (Spatial stability): El grado en el que los datos se mantienen estables entre diferentes fuentes de datos.

- Contextualización (Contextualization): El grado en el que los datos están correcta y óptimamente anotados con el contexto en el que han sido adquiridos.

- Valor predictivo (Predictive value): El grado en el que los datos contienen información relevante para propósitos de procesos de toma de decisiones específicas.

- Fiabilidad (Reliability): El grado de reputación de las instituciones y los "stakeholders", proporcionando un ranking de puntuaciones de aquellos involucrados en la adquisición de los datos.

2.2. Ejes

Las diferentes dimensiones anteriormente mencionadas, se pueden aplicar a distintos ejes propios de los datos, de esta manera se puede precisar más, al medir la calidad tanto en pequeñas porciones de los datos, como en el conjunto global de todos ellos. Los distintos ejes a los que se pueden aplicar las dimensiones de calidad son:

- Registro (Registry): El sujeto de hecho, un paciente, contacto, episodio etc. Sería equivalente a las filas en una matriz en una vista minable.
- Atributo (Attribute): El conjunto de las diferentes variables del conjunto de datos. Sería equivalente a las columnas de una matriz.
- Valor (Value): Representa cada dato individualmente, es donde la información está contenida. Sería equivalente a cada celda de un matriz.
- Dataset: El conjunto de todos los valores, es el conjunto de datos al completo.
- Tiempo (Time): Permite la comparación de datos entre diferentes periodos de tiempo, puede ser entendido como un caso especial de un eje atributo de contexto que indica cuando el dato ha sido registrado.
- Fuente (Source): Permite la comparación de datos entre diferentes fuentes o lugares donde el dato ha sido recogido. Este eje también se puede ver como un caso especial de un atributo de contexto que indica la fuente de datos.

2.3. Dimensiones y ejes implementados

En el proyecto que se presenta se han conseguido implementar en el servicio web las siguientes dimensiones, accesibles desde la interfaz web también desarrollada. Los ejes han sido igualmente implementados para las citadas dimensiones, exceptuando los de Tiempo y Fuente.

2.3.1. Completitud

Como se ha introducido, la completitud mide el grado en el que datos relevantes han sido registrados en el conjunto de datos. Concretamente en la implementación realizada, la completitud devuelve el porcentaje de elementos completos o rellenados en el eje seleccionado.

La implementación está realizada de manera que se recorren los elementos correspondientes al eje donde se realiza la medida, comprobando cuantos están efectivamente completos, devolviendo un valor entre 0 y 1 correspondiente al porcentaje de elementos completos.

El comportamiento entre los distintos ejes es similar, en el caso del eje dataset el resultado obtenido será el porcentaje total de datos completos en el conjunto global de datos. En el caso del los ejes atributo y registro, el resultado será el porcentaje total del eje correspondiente para cada uno de los atributos o registros contenidos en el dataset. En el caso del eje valor, el resultado será una matriz de ceros y unos, correspondientes a los valores vacios y completos respectivamente, hallados en la totalidad del conjunto de datos.

2.3.2. Consistencia

La consistencia proporciona información sobre como de consistentes son los datos de los que disponemos, cumpliendo con una serie de reglas proporcionadas y dependiendo si los datos de un atributo son numéricos o categóricos. Estas reglas dependen del contexto en el que se estén midiendo los datos, y aunque se pueden definir reglas muy generales, cada contexto requiere de su conjunto de reglas personalizado. Por este motivo se ha decidido incluir un modulo empleando programación declarativa, para poder evaluar las reglas proporcionadas pudiendo utilizar distintos archivos o conjuntos de reglas, dependiendo el contexto, y dejando la posibilidad de que en el futuro sea el usuario final el que defina y utilice sus propias reglas personalmente. Para este efecto, se ha utilizado la plataforma de reglas de negocio Drools, cuyos detalles se discutirán más adelante, para poder integrar este módulo de programación declarativa dentro del proyecto Java que utiliza programación imperativa.

Debido al grado de personalización, en el proyecto se han incluido tres reglas, para poder realizarse pruebas que se detallan más adelante:

- No es un número: Como los diferentes valores pueden ser de tipo numérico o categórico, esta regla se encarga de comprobar si el usuario ha declarado que un atributo está formado por valores numéricos, entonces el valor efectivamente se trata de un número, devolviendo en caso contrario el código NAN|(valor), donde (valor) contiene el valor que ha provocado que se dispare la regla.

- Menor que cero: De manera análoga a la regla anterior, primero se comprueba que efectivamente es un número el valor que se está evaluando, y a continuación, se comprueba que el valor es menor que cero, devolviendo el código LTZ|(valor) donde (valor) contiene el valor que ha provocado el disparo de la regla.

- Hombre embarazado: Esta regla se encarga de evaluar que si un atributo categórico contiene la cadena “man”, refiriéndose a hombre, no existe en el mismo registro otro atributo categórico que contenga la cadena “pregnant”, refiriéndose a embarazo, haciendo cumplir la restricción biológica de que no pueden haber hombres embarazados, y devolviendo el código MPG cuando se ha detectado alguno. En el caso de esta regla, al relacionar dos atributos distintos dentro de un registro, solo tiene sentido aplicarla cuando estamos midiendo la consistencia de los datos a nivel de registro.

Los diferentes ejes se han tratado igual que en el apartado de completitud, con la diferencia de que se crea una base de hechos distinta para cada atributo o registro según el caso, de esta manera se mantiene el aislamiento entre distintos ejes, creándose una única base de hechos en el caso del eje dataset.

3. Tecnologías utilizadas

En este apartado se detallan las tecnologías utilizadas para la realización del proyecto, junto con los motivos que han ayudado a la elección de las distintas herramientas.

3.1. Eclipse JEE y EMF

El entorno de programación Eclipse, proporciona un conjunto de herramientas de código abierto para programación, con soporte para el lenguaje de programación Java, siendo de especial utilidad para la depuración del código y de sus funciones, por lo que se ha elegido como entorno de desarrollo para el proyecto.

Eclipse engloba distintas distribuciones que incluyen de manera predeterminada módulos o “plugins” para la realización de tareas especificas. En la realización del proyecto se han utilizado dos distribuciones de eclipse distintas, Eclipse Modeling Framework (EMF)[3] y Eclipse JavaEE (EJEE)[4].

En la primera parte del proyecto se ha utilizado EMF, para diseñar la estructura de clases que contiene los datos y los métodos empleados para calcular la calidad de los datos. Así como para generar el código asociado al diagrama UML creado (figura 1), ya que el generador de código emplea un patrón de diseño de tipo factoría [7], útil en la posterior creación y gestión de instancias.

Tras la generación de manera automática de las clases, se pasó a utilizar EJEE para la programación de las clases y métodos generados. Además, se utilizó el servidor TOMCAT incluido en esta distribución para realizar distintas pruebas durante el

3 http://www.eclipse.org/modeling/emf/

4 http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/lunar

desarrollo, adicionalmente se procedió a la inclusión del plugin de Vaadin para realizar el desarrollo de la interfaz en este entorno, y el módulo de Drools para desarrollar las reglas de prueba y el apartado de programación declarativa.

3.2. Jersey

El proyecto realizado tiene como requerimiento que debe ser un servicio web de tipo REST, para ello se ha optado por utilizar el framework Jersey RESTful Web Services[5], que proporciona las herramientas necesarias para facilitar la comunicación entre cliente y servidor.

Los motivos para la elección de Jersey han sido:

- Jersey es un toolkit para Java, lenguaje en el que se desarrolla el proyecto.
- La propia definición del proyecto incluye el requerimiento de que la aplicación desarrollada tiene que ser un webservice, por lo que se descarta el uso de Java-web Servlets y Java Server Pages (JSP).
- La licencia de Jersey es OpenSource, por lo que permite su utilización sin necesidad de comprar ninguna licencia.
- Cuenta con una documentación amplia. Tanto es su propia web, como en forma de tutoriales en internet.
- Jersey es sencillo de utilizar una vez configurado, y permite realizar de manera simple la comunicación entre cliente y servidor.
- Recomendación por parte de miembros del grupo de investigación, al haber dado buenos resultados en proyectos anteriores.

Tras incluir los archivos .jar correspondientes a Jersey en el proyecto eclipse, este se encarga totalmente de realizar la comunicación y las peticiones POST empleadas en el proyecto.

3.3. Vaadin

Vaadin[6] es una herramienta para construir interfaces gráficas de usuario (GUI) mediante código Java, y accesibles desde cualquier navegador web sin que el usuario tenga que instalar nada, ya que utiliza html5.
Los motivos de la elección de Vaadin han sido:

- Framework que utiliza Java, el lenguaje utilizado en el proyecto.
- Vaadin es software libre, distribuido bajo licencia Apache 2.0.
- Vaadin dispone de un plugin para eclipse, simplificando su instalación y proporcionando herramientas para el diseño.

5 https://jersey.java.net/

6 https://vaadin.com/home

- Vaadin crea la interfaz como una aplicación web independiente, lo que permite tener la interfaz separada del webservice al que accede, pudiendo estar en servidores distintos.
- Experiencias previas positivas del grupo de investigación con el framework.

Para la creación de la interfaz, se creó un proyecto eclipse independiente con la instalación del plugin y se añadieron las bibliotecas .jar correspondientes. Cabe destacar que no se utilizaron plugins adicionales para Vaadin en la creación de la interfaz. El estilo que utiliza la aplicación para el diseño de botones, color de fondo, forma de los distintos elementos etc. Está disponible en la propia web de Vaadin, y tiene como nombre "Dawn".

3.4. Drools

Drools[7] es un sistema de gestión de reglas de negocio (BRMS), que utiliza una implementación avanzada del algoritmo RETE. La versión utilizada en el proyecto es la "6.0 Drools Expert", para aplicar, añadir y evaluar reglas desde un fichero de reglas para Drools.
Los motivos para la elección de Drools han sido:

- La necesidad de incorporar un módulo de programación de tipo declarativa, o basada en reglas, para evaluar reglas de consistencia.
- Drools es OpenSource, y está publicado bajo licencia Apache (ASL).
- La sintaxis empleada es similar a otras ya conocidas, como las empleadas en CLIPS y JESS.
- La facilidad de incorporación y utilización en el proyecto.
- La amplia documentación y tutoriales disponibles en internet.
- Oportunidad de conocer y emplear un nuevo BRMS de código libre.

Las bibliotecas necesarias para el uso de Drools se han incluido en el proyecto, así como el archivo de reglas generado, almacenado en la carpeta de "Java Resources", delegando las operaciones necesarias para la evaluación de reglas a la propia herramienta.

7 http://drools.jboss.org/

4. Diseño del servicio para control de calidad

Debido a la escalabilidad del software a diseñar, se decidió separar el desarrollo en tres proyectos distintos en eclipse, de tal manera que se pudieran reutilizar partes del software en proyectos futuros, o mejorar alguna parte con carácter independiente del resto de proyecto. Por un lado, se desarrollo el código del proyecto DQV con carácter "offline", implementando las estructuras de datos y métodos necesarios para el funcionamiento y posterior uso como biblioteca Java ".jar". A continuación, se desarrolló el servicio web, que contiene la biblioteca generada para responder a las peticiones recibidas. Y por último, se dió forma a una interfaz web mediante la cual se pueden realizar peticiones y visualizar los resultados. Todos los proyectos de eclipse se desarrollaron añadiendo funcionalidades de manera incremental y comprobando el funcionamiento mediante pruebas, tanto de tipo individual, como en el conjunto de las aplicaciones. Cabe señalar que todos los proyectos se realizaron utilizando la versión de Java JRE6, aunque en un principio se había optado por utilizar JRE7, se decidió realizar los cambios para utilizar la versión 6, por motivos de compatibilidad. Los detalles del desarrollo y funcionamiento se detallan a continuación.

4.1. Proyecto DQV Java

El proyecto llamado en eclipse "DQV_project" contiene todos los métodos utilizados para medir la calidad de los datos y las estructuras que contienen estos datos.

Al realizar la primera aproximación del diseño de la aplicación, se pudo observar que, al tener seis ejes distintos aplicables a nueve dimensiones, se encuentra la problemática de complejidad al haber muchas rutinas distintas a mantener, por lo que habría que buscar una solución que facilitara la escalabilidad y el mantenimiento. Para evitar un alto número de clases y métodos que daría la combinatoria de dimensiones y ejes, se ha optado por la implementación de un patrón de diseño estrategia, (strategy pattern)[7] [8] donde se hereda de clases abstractas a clases más especificas que contienen sus propios métodos para preparar las estructuras y realizar los cálculos. Como resultado de estudiar los requisitos funcionales y la estrategia elegida, se ha implementado la arquitectura mostrada en la figura 1.

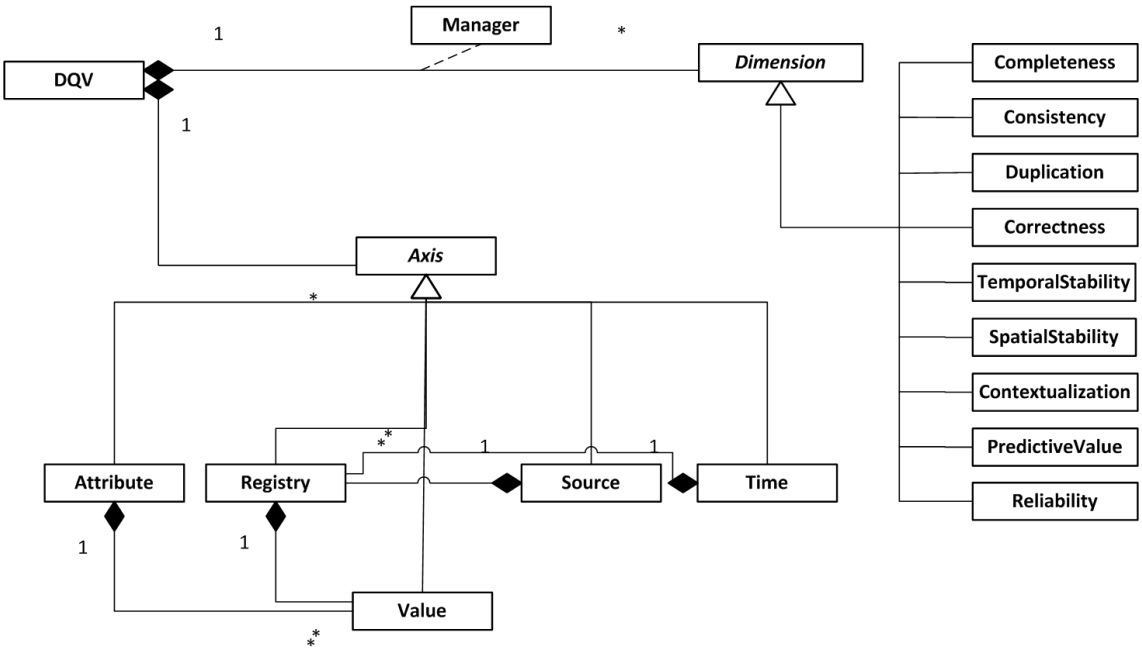

Figura 1. *Diagrama UML del sistema.*

El dataset quedarepresentado de manera implícita en la figura 1 como el conjunto de clases de ejes específicos, que tienen como nombre "Attribute", "Registry", "Source", "Time" y "Value", donde representan los ejes llamados atributo, registro, fuente, tiempo y valor. Todos ellos heredan desde una clase común llamada "Axis" (eje). Cada eje de tipo atributo, registro, fuente y tiempo tiene asociado un conjunto de clases de tipo Value (valor), donde se encuentra contenido el valor del dato. De esta manera, tenemos asociados todos los datos en las distintas clases implementadas como listas de objetos dinámicos, por lo que conceptualmente tendríamos una matriz de valores enlazados entre sí mediante los registros, los atributos, o enlazadas mediante todos los valores del dataset, ya que la clase DQV contiene todos los conjuntos de estas clases.

La clase DQV también contiene los métodos necesarios para realizar los cálculos, para ello se han definido una serie de clases correspondientes al método utilizado para calcular las diferentes dimensiones. Estas clases, como "Completeness" o "Concistency" heredan de una clase abstracta "Metric", tal y como se describe en el patrón de diseño estrategia, y dentro de cada clase correspondiente, se realizan los cálculos específicos para la dimensión en concreto. Estos cálculos están detallados en el apartado 2.3 para las dimensiones implementadas.

Para manejar la complejidad de tener seis ejes distintos para realizar cálculos en cada dimensión, se ha decidido hacer uso de una clase asociada llamada "Manager". Esta clase es la encargada de preparar la estructura y tratar las distintas maneras de calcular cada dimensión con el correspondiente eje, reusando el código existente en cada métrica, y almacenando los resultados.

Para realizar las llamadas que calculan la calidad en los datos de la dimensión seleccionada, hay que invocar al método startDQV(string xml), donde hay que pasarle como parámetro una cadena de texto con el XML que contenga la información de la dimensión y el eje a calcular, así como los datos y el tipo de cada atributo, el XML

debe cumplir con el esquema incluido en el anexo 1 para el correcto funcionamiento de la aplicación, que se puede ver de manera esquemática en la figura 2.

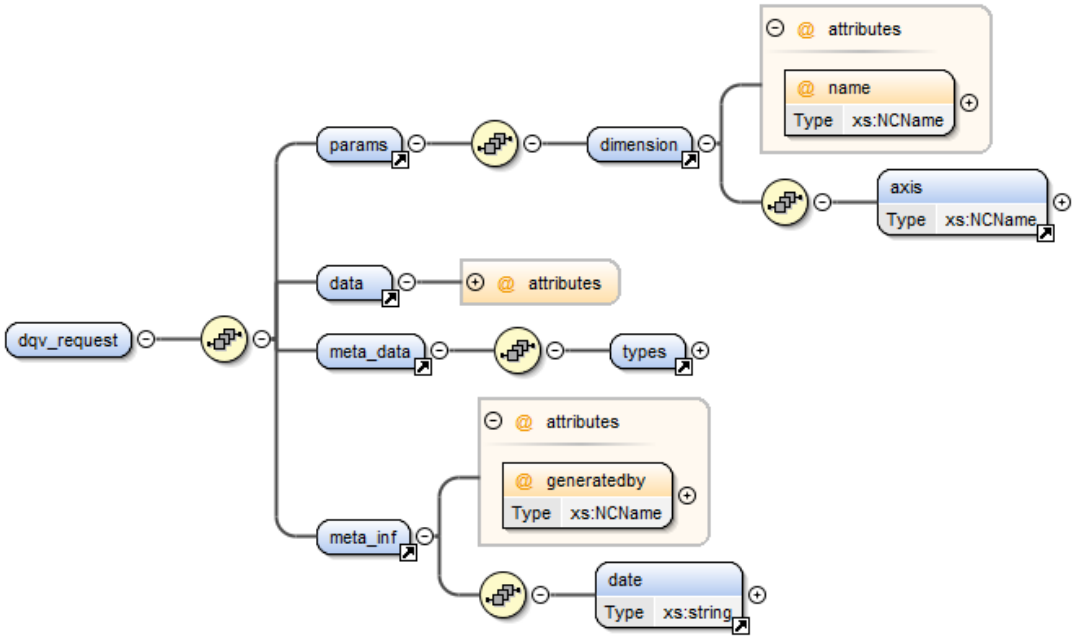

Figura 2. *Esquema XML para las peticiones.*

Como resultado de la llamada, se obtendría un objeto de tipo Result (resultado), que contiene una lista de cadenas de texto con los distintos resultados obtenidos, dependiendo de la dimensión y eje seleccionados.

Si en el futuro se quisiera añadir la implementación de una nueva dimensión, o modificar el cálculo de una ya existente, únicamente habría que rellenar el código correspondiente a la métrica de la dimensión elegida, en lugar de replantearse la estructura global o modificar otras clases que se viesen afectadas si se hubiera seguido una arquitectura clásica, por lo que se aprecia la facilidad para el mantenimiento de la aplicación.

4.2. Servicio Web DQV

El servicio web está incluido en el proyecto eclipse llamado “dqv_service” y contiene el servicio web en su totalidad, listo para ser publicado sobre un servidor web. En este caso, y como se ha indicado en las contribuciones, se ha optado por desplegar el servicio web en un servidor de tipo TOMCAT Versión 7. Dentro de este proyecto está incluida la biblioteca generada a partir del proyecto anterior, lo que permite la creación de la clase DQV necesaria para la utilización de los métodos asociados para el cálculo de calidad. El contexto del servicio web, que contiene la biblioteca con los métodos para el cálculo, se puede apreciar en la Figura 3, donde se observan las interacciones tanto de la interfaz creada, como de terceras aplicaciones.

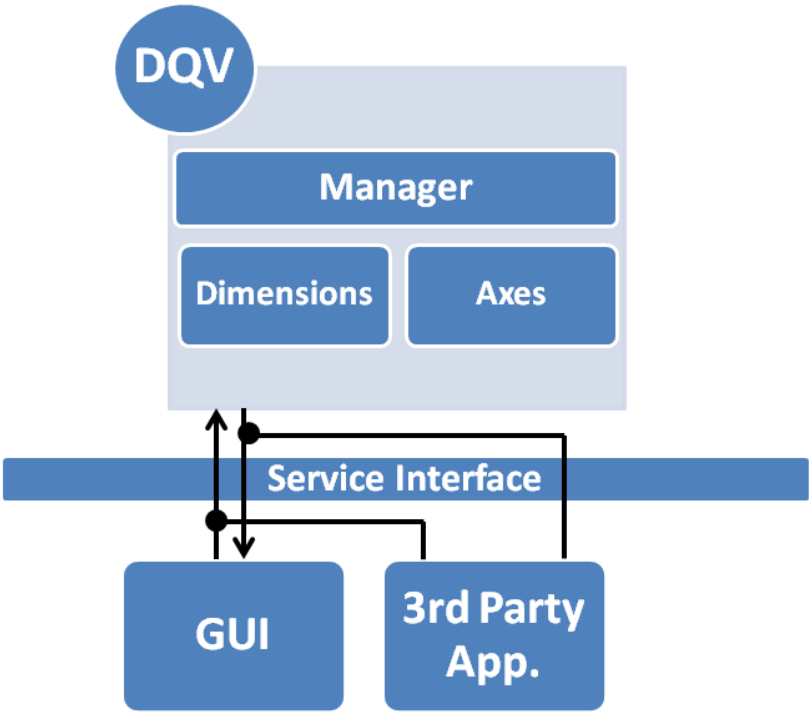

Figura 3. *Contexto del servicio web.*

La estructura interna del servicio web está muy simplificada gracias al uso de Jersey, ya que basta solamente con las bibliotecas incluidas y una clase para responder a las peticiones recibidas. Principalmente, el servicio web responde a peticiones de tipo POST, ya que el servicio web recibe un XML con la petición, como se detalla en el apartado anterior, y se responde con otro archivo XML con los resultados obtenidos. Para realizar la conversión de los resultados a un archivo de tipo XML, se hace uso de una clase auxiliar creada para tal efecto. Esta clase auxiliar utiliza JAXB para realizar una conversión automática del objeto resultado en un archivo XML, mediante el uso de anotaciones java (Java annotations). Se puede ver un extracto de la clase empleada con el java anotado en la figura 4. El esquema XML que genera esta clase se puede consultar en el ANEXO 2.

```
 9 @XmlRootElement
10 public class MyXmlResult {
11     @XmlElement(name="dimension")
12     public String dimension;
13     @XmlElement(name="axis")
14     public String axis;
15
16     @XmlElement(name="rows_dataset")
17     public int rows;
18     @XmlElement(name="columns_dataset")
19     public int columns;
20
21     @XmlList
22     public List<String> values;
23
```

Figura 4. *Java anotado para la conversión a archivos XML.*

El funcionamiento es igualmente sencillo, tras recibir una petición de tipo POST junto con el XML correspondiente, se crea una instancia de la clase DQV incluida como biblioteca y se devuelve el resultado generado, para lo cual se hace uso de la clase auxiliar para crear el fichero XML con JAXB y enviarlo a quien ha realizado la petición.

Adicionalmente, se ha incluido en el webservice la respuesta a peticiones de tipo GET para poder comprobar que el servicio web está funcionando correctamente. Esta petición se responde con un texto plano indicando que el servicio web está trabajando, y para realizar el test basta con realizar una petición GET a la dirección base del webservice añadiendo "/test" al final de la dirección, como se aprecia en el apartado 1.3 de contribuciones. En cuanto al funcionamiento de la petición de test del servicio, simplemente se devuelve la frase "I'M WORKING", quedando el texto en pantalla si esta consulta se hace desde un navegador web.

4.3. Interfaz Web DQV

La interfaz web ha sido realizada para poder utilizar el servicio web desde cualquier navegador. Tal y como se ha indicado previamente, la interfaz esta realizada en su totalidad mediante Vaadin, y está contenida en el proyecto de eclipse llamado "DQV_GUI", pudiéndose exportar y desplegar en un servidor directamente como se ha indicado en el apartado contribuciones.

La interfaz está separada en dos vistas diferentes, para simplificar la implementación y para mantener una filosofía modular que facilite el mantenimiento. Estas dos vistas se llaman "Firstframe" y "Resultframe", y se detallarán a continuación. Adicionalmente, se emplea una clase llamada "Dqv_guiUI" para realizar la configuración y una clase auxiliar llamada "Xmlbuilder", detallada más adelante, para la construcción de la petición XML.

4.3.1. Firstframe

La vista llamada "Firstframe" es la primera que verá el usuario al utilizar la interfaz del servicio web. Esta vista contiene los botones necesarios para seleccionar la dimensión y el eje donde quiere calcular los datos. También contiene la estructura de datos necesaria para almacenar las opciones seleccionadas, y los manejadores de los eventos que se producen al seleccionar las distintas opciones, como cargar los datos tras realizar la subida, o seleccionar la dimensión deseada.

Hay que destacar el evento que se produce tras subir el archivo del usuario, que provoca la carga de los datos en memoria, y el que se produce tras pulsar sobre el

botón “calculate”, ya que inmediatamente después se procede a la construcción del archivo XML con la petición a realizar, y se encarga de la comunicación con el webservice, utilizando una petición de tipo POST para enviar los datos y recibir los resultados, para acto seguido cambiar a la vista de resultados.

El usuario puede subir el archivo CSV mediante la opción de “examinar”, seleccionando la ruta donde se encuentra, y pulsando sobre el botón de “upload”, para realizar la subida de datos. En ese momento se cargaran unas pocas líneas del archivo en un pequeño cuadro situado a la derecha, para poder comprobar visualmente el archivo subido, y aparecerán una lista de selectores por cada atributo detectado en el fichero, para poder elegir el tipo de dato, ya sea categórico o numérico que contiene el atributo seleccionado.

Como normalmente la primera línea del fichero contiene metadatos, como el nombre de las columnas, existe la opción de saltarse la primera línea del archivo para realizar los cálculos, seleccionando la opción “skipfirst” en la interfaz. Tras seleccionar la dimensión y el eje donde se quiere realizar la medida de calidad, si se pulsa sobre “calculate” se realizan los cálculos y se pasa a la vista de resultados, que se detalla en el próximo apartado.

4.3.2. Resultframe

La vista llamada “Resultframe” contiene la estructura necesaria para poder mostrar los datos de manera visual al usuario. En esta vista, tenemos los botones que nos permiten volver a la vista anterior, y descargarnos el archivo de resultados generado.

Tras la llamada que cambia la vista a “Resultframe”, se procede a cargar los datos recibidos desde el archivo XML de resultados a memoria. Dependiendo de la dimensión y del eje calculado, los resultados se muestran de distinta manera para facilitar su comprensión, adaptando la tabla de resultados al número de filas y columnas que sean necesarias

4.3.3. Xmlbuider

Para la construcción del archivo XML que se enviará como petición del webservice, se ha optado por la creación de una clase auxiliar llamada “Xmlbuilder”.
El funcionamiento de la clase es sencillo, y basta con pasarle las distintas variables cargadas en memoria, para recibir el archivo XML que respeta el esquema que utiliza el servicio web (Figura 2 y Anexo 1). Internamente hay distintos métodos para la creación de este fichero, similares entre sí pero diferenciados para estructurar y simplificar la creación del fichero final.

5. Resultados

A continuación, se presentará un caso de uso de la aplicación y una prueba de rendimiento. Para ambos casos se ha utilizado un fichero de datos en formato CSV, con datos sintetizados a partir de datos reales para poder probar la herramienta.

5.1. Caso de uso

Para la presentación del caso de uso hemos utilizado la interfaz web que accede al servicio web desarrollado. Tras acceder a la dirección mediante navegador web, podemos observar la interfaz vacía, como se muestra en la figura 5.

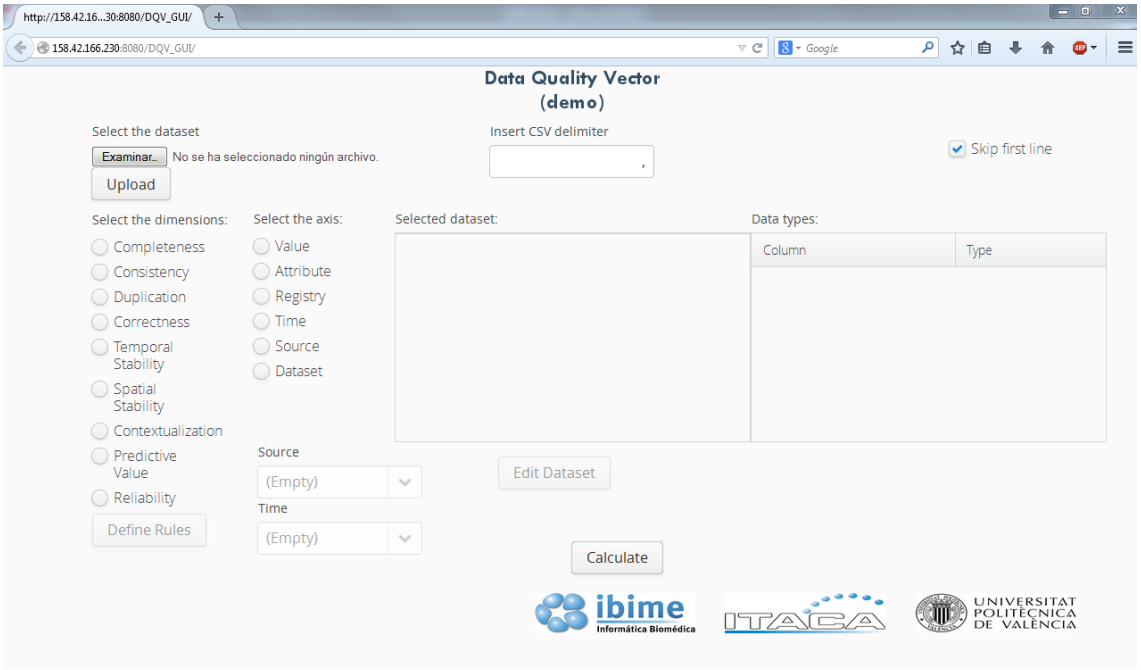

Figura 5. ***Interfaz Grafica de Usuario.***

Tras pulsar sobre el botón de examinar, elegimos los datos que queremos analizar, el nombre del fichero seleccionado aparecerá en pantalla, como se puede observar en la figura 6.

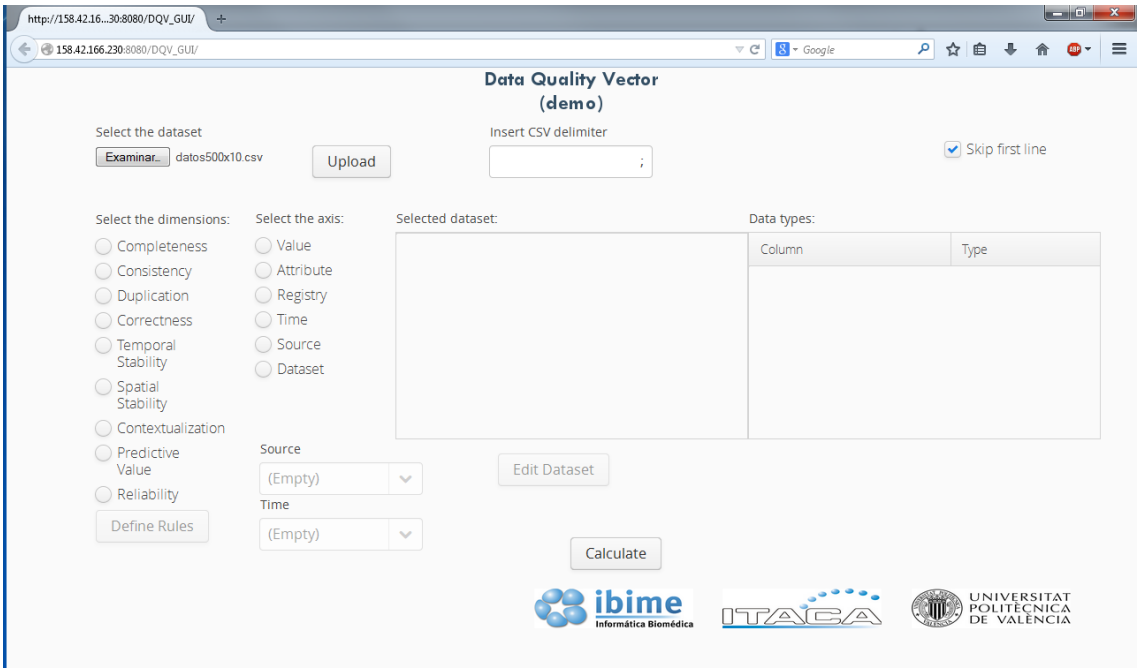

Figura 6. *GUI con fichero seleccionado.*

El siguiente paso es subir los datos seleccionados, tras lo cual podemos observar como la intefaz se rellena automáticamente en la figura 7.

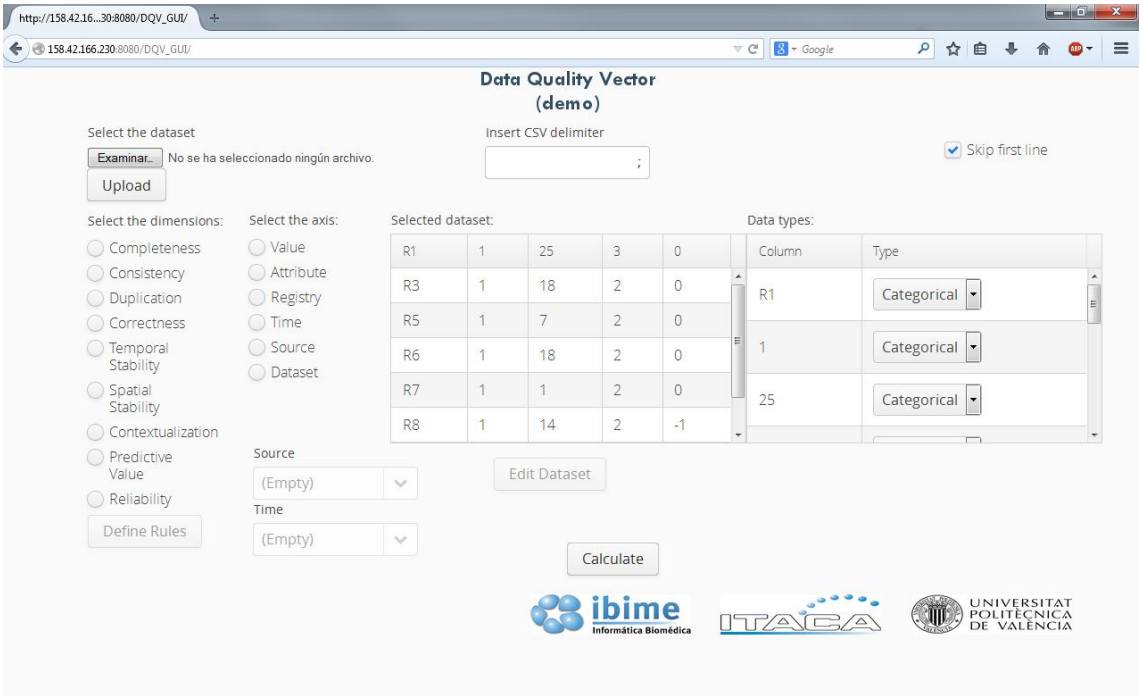

Figura 7. *GUI con datos cargados.*

En la tabla de la izquierda de la figura 7 se pueden observar las primeras líneas del archivo subido, y en la parte derecha, el selector de los tipos de datos correspondientes a los atributos.

Tras seleccionar la dimensión y el eje deseados, quitamos la marca a la caja de "skip first line", ya que nuestro archivo no contiene metadatos en la primera línea. Además, y aunque en esta dimensión no es necesario, se ha marcado uno de los atributos como datos de tipo numérico como se observa en la figura 8.

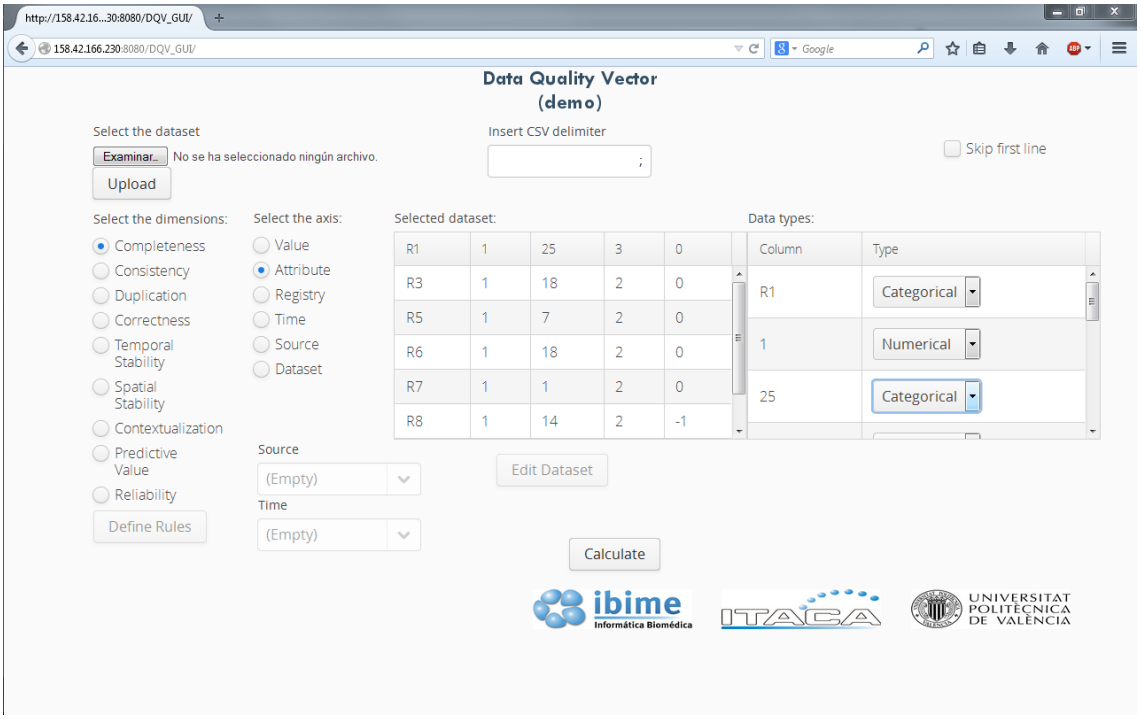

Figura 8. *GUI con opciones seleccionadas.*

Una vez rellenados los datos necesarios, solo queda pulsar sobre el botón "Calculate" para obtener los siguientes resultados, visibles en la figura 9.

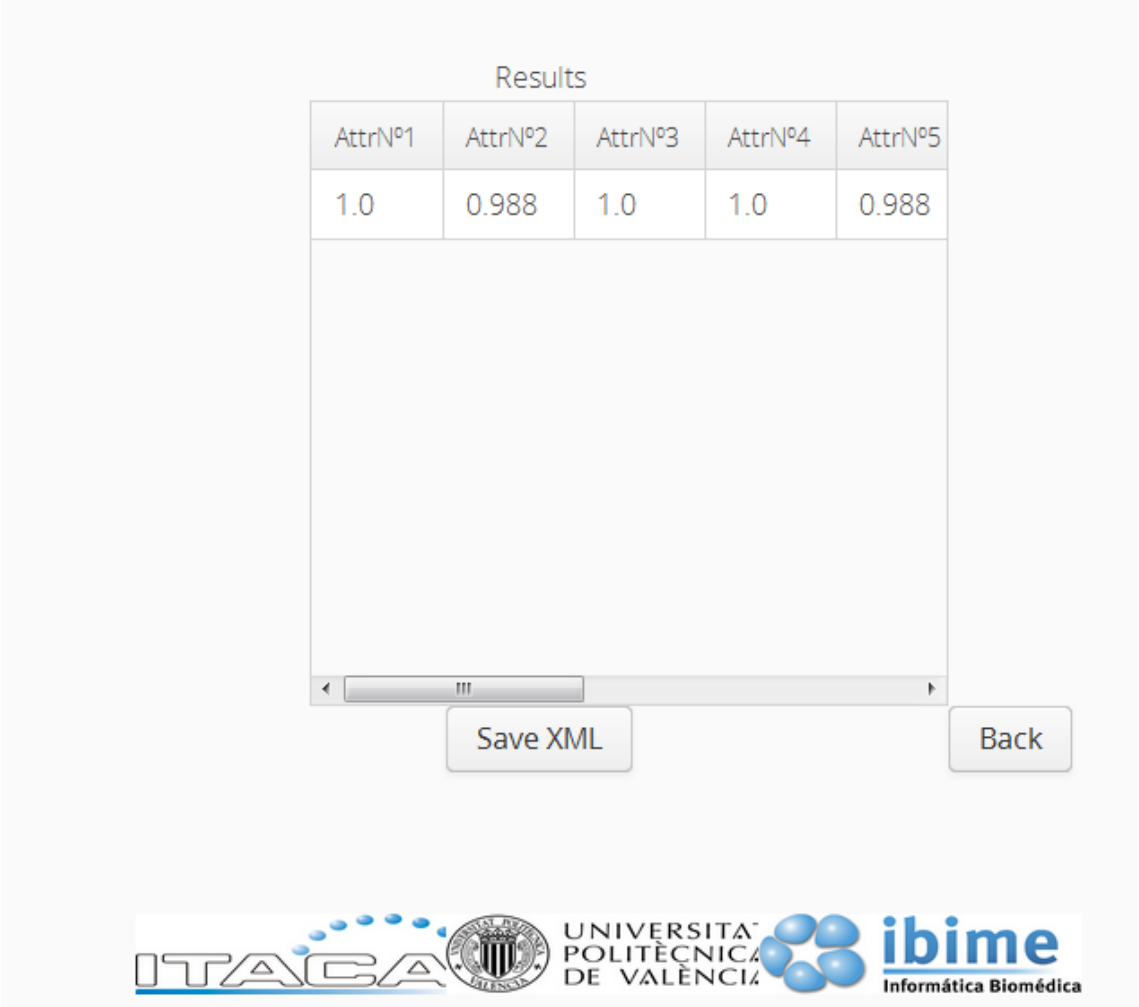

Figura 9. ***Pagina de resultados de completitud a nivel de atributo.***

En el resumen de los resultados de la figura 9 se aprecia como el atributo número uno está completo al 100%, mientras que el número dos, se encuentra al 98,8%.

Si pulsamos sobre el botón de "Back", volveremos atrás y tendremos cargados los datos que hemos utilizado, en este momento podemos seleccionar otra dimensión o eje para medir su calidad como se puede apreciar en la figura 10.

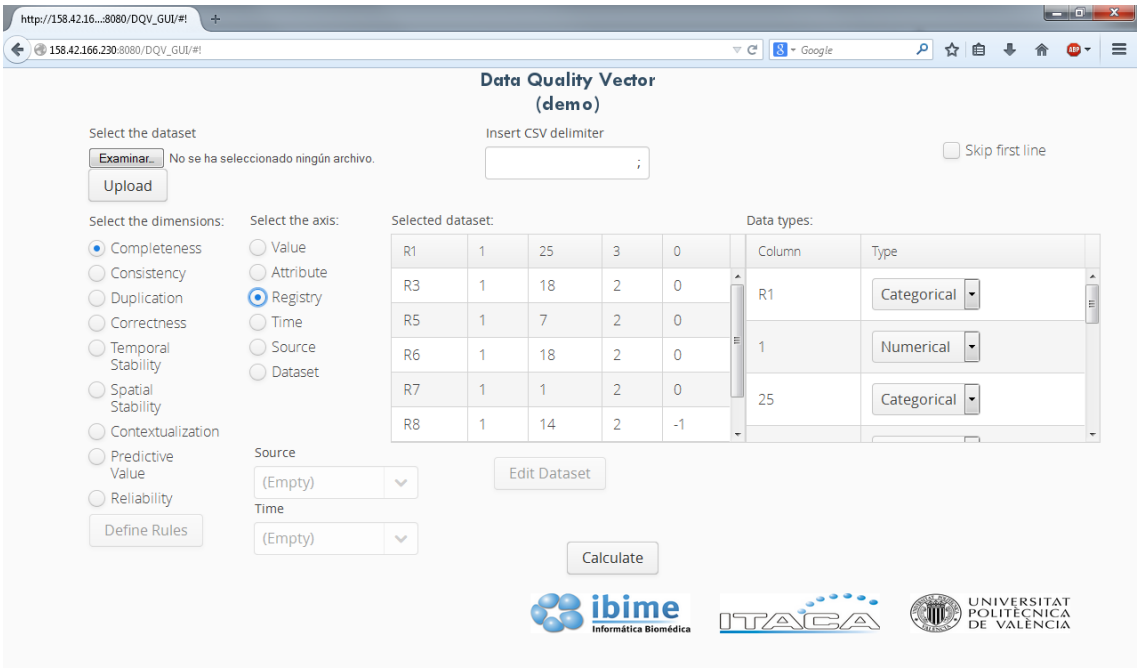

Figura 10. *GUI con los datos previamente cargados.*

Tras volver a solicitar el cálculo, se obtienen los resultados detallados en la figura 11.

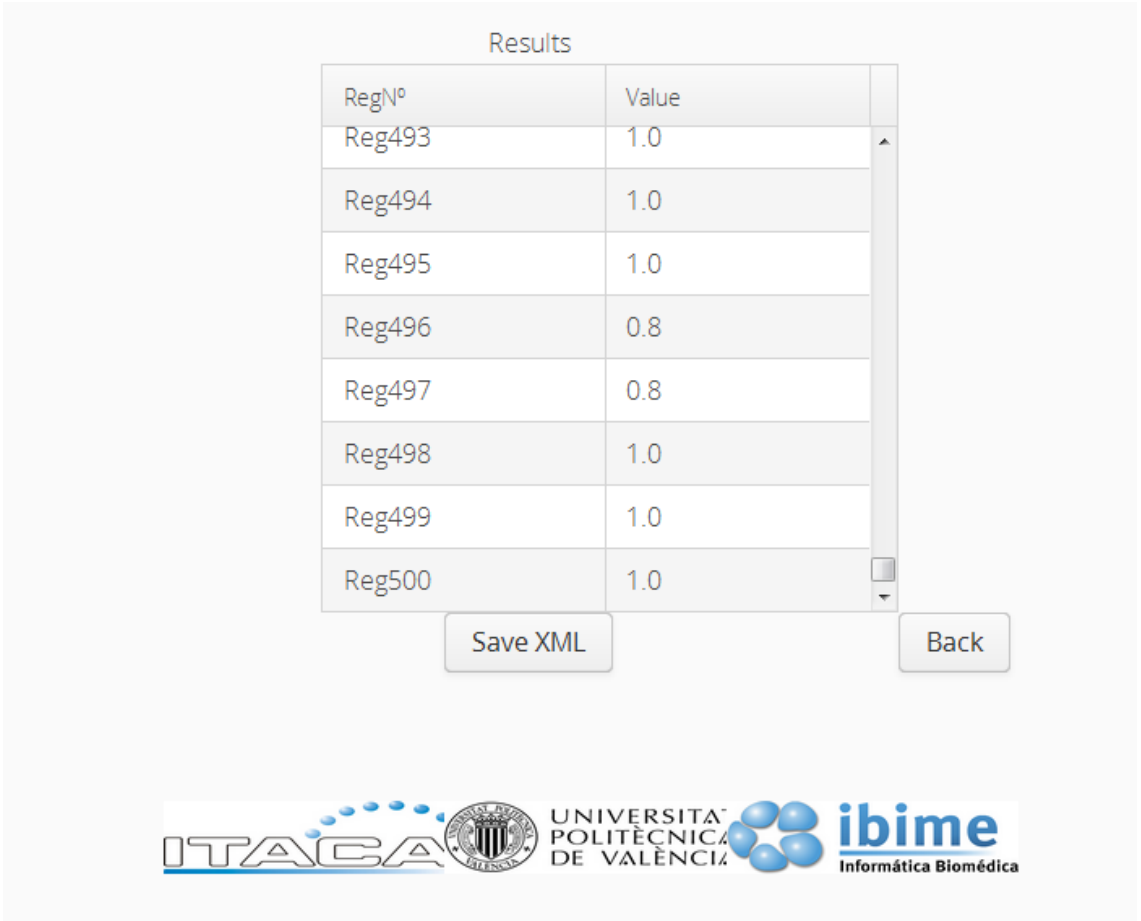

Figura 11. *Resultados de completitud a nivel de registro.*

En la figura 11 se puede apreciar como registros 496 y 497 están completos solo al 80%. Tras esto, podríamos guardar los resultados en formato XML o volver atrás para realizar más cálculos.

5.2. Rendimiento

Para la realización de las pruebas de rendimiento, se ha empleado un ordenador con procesador Intel Core 2 Duo, modelo T8100 con dos núcleos a 2.10GHz y 4GB de memoria RAM. Tanto el servicio web como la interfaz gráfica se desplegaron sobre el mismo servidor TOMCAT versión 7.0, y se midieron los tiempos desde el envío de la petición al webservice, hasta la recepción de la misma.

Se ha realizado una prueba de rendimiento por cada dimensión implementada, utilizando para ambos caso el eje de registro, ya que por su codificación es el que provoca más llamadas intermedias, y debido a que generalmente los datos contienen más registros que atributos, nos encontramos ante el peor caso que se puede dar en la aplicación.

Para ambas pruebas se han utilizado los mismos datos, y tratando de replicar la situación que se da al tener más filas que columnas, se ha utilizado un número de columnas fijo (10 atributos) y un número de registros que será de 500, 1.000, y 10.000 en cada caso. Primero se realizaron individualmente diez ejecuciones para obtener el tiempo medio de petición, y después se realizaron automáticamente en bucle 100 peticiones al servicio, midiendo también el tiempo medio por ejecución.

5.2.1. Rendimiento de la dimensión completitud

Tras realizar las ejecuciones y medidas correspondientes, hemos obtenido la siguiente gráfica:

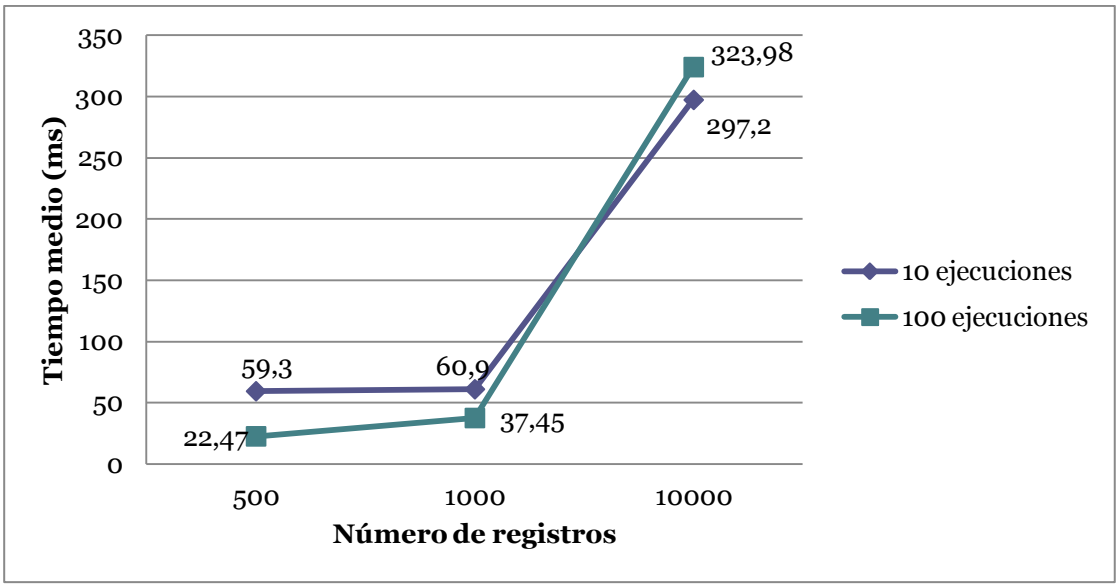

Figura 12. ***Rendimiento de la dimensión completitud.***

Como se puede observar en la figura 12, al realizar las peticiones al servicio utilizando el bucle, se obtiene un mejor rendimiento, pero al tratarse de tiempos del orden de 50 milisegundos o menos, suponemos que esta variabilidad se produce debido al reparto de recursos y asignación de prioridades que realiza el sistema operativo, obteniendo más prioridad la tarea de realizar cien medidas al requerir más tiempo de cómputo. El mismo efecto también se puede observar al aumentar el número de registros de 1.000 a 10.000, tras realizar ejecuciones individuales, ya que este último resultado es sensiblemente menor a 609 ms, que sería el tiempo correspondiente a la ejecución de 1.000 registros multiplicado por diez. Adicionalmente, observamos como el tiempo apenas varía al doblar los registros de 500 a 1000. Por otro lado, al multiplicar por diez el número de registros se observa como el tiempo aumenta linealmente, siendo incluso ligeramente inferior que el tiempo correspondiente a 1.000 registros multiplicado por diez.

5.2.2. Rendimiento de la dimensión consistencia

Para realizar las pruebas se definieron los tipos de datos correspondientes a los atributos del dataset, definiendo la mitad de ellos como de tipo categórico y la otra mitad de tipo numérico, para poder evaluar las reglas que afectan a un tipo y a otro. Tras realizar las ejecuciones y medidas correspondientes, hemos obtenido la siguiente gráfica:

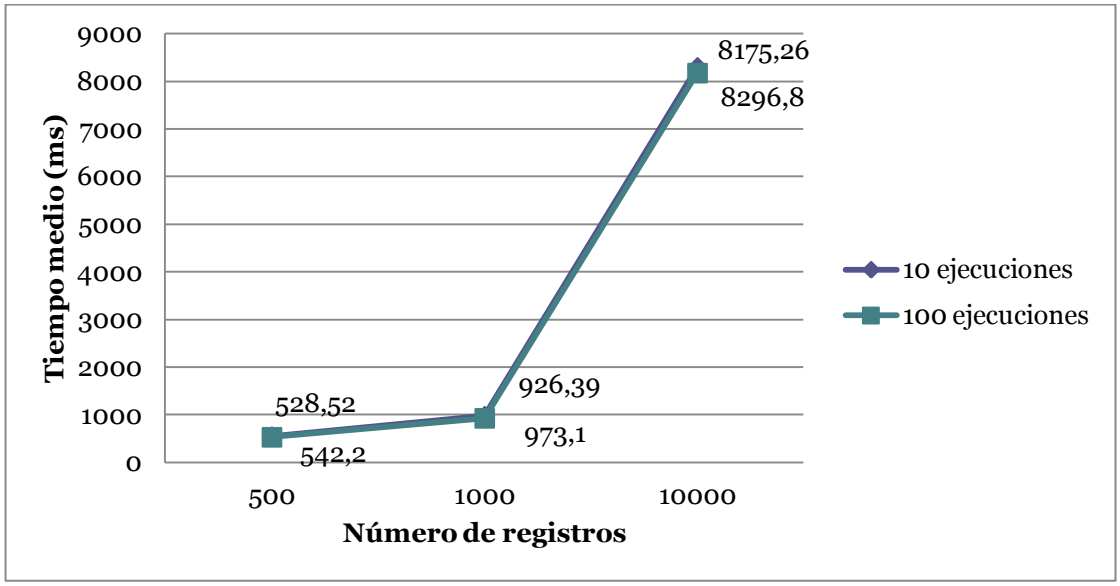

Figura 13. ***Rendimiento de la dimensión consistencia.***

Como se puede observar en la figura 13 y realizando una comparación con la figura 12, se observa que los tiempos aumentan significativamente al utilizar la dimensión de consistencia. Los tiempos de rendimiento medio varían desde el medio segundo de los 500 registros, hasta los ocho segundos al utilizar 10.000 registros, esto está provocado por la utilización de programación declarativa, y tras la ejecución de los algoritmos internos de Drools. Como se puede observar las diferencias internas entre

las ejecuciones individuales o en bucle no son significativas, ya que como se ve las gráficas están prácticamente solapadas, siendo los valores inferiores los correspondientes a las medias de las ejecuciones individuales, los que son ligeramente inferiores por el mismo motivo comentado en el apartado anterior. Igualmente, se observa un crecimiento lineal del tiempo requerido para responder a la petición, por lo que se ven cumplidos los requisitos de rendimiento.

6. Conclusiones

Tras la realización del proyecto y la elaboración de la presente memoria, podemos destacar como conclusiones las siguientes:

- La utilización de dimensiones permite la detección de fallos en distintos aspectos de calidad de datos.
- El uso de ejes, en conjunción con las dimensiones, permite analizar a distintos niveles los datos, proporcionando información detallada de su calidad.
- Las tecnologías utilizadas tienen en su totalidad licencias de código libre que permiten su utilización gratuita.
- Ha quedado demostrada la factibilidad de implementación de dimensiones tanto de conteo como las basadas en reglas.
- El servicio web desarrollado tiene rendimiento lineal para las dos dimensiones implementadas.
- El servicio web desarrollado resulta accesible desde cualquier navegador gracias a la interfaz creada a tal efecto, y a terceras aplicaciones mediante peticiones desde internet.
- El objetivo de escalabilidad ha sido respetado durante todo el desarrollo, utilizando estrategias de diseño como el patrón estrategia, y técnicas orientadas al uso modular para lograrlo.

7. Referencias

[1] N. G. Weiskopf and C. Weng. Methods and dimensions of electronic health record data quality assess-ment: enabling reuse for clinical research. J Am Med Inform Assoc, 20(1):144–151, Jan 2013.

[2] Prakash M. Nadkarni and Randolph A. Miller, Service-oriented Architecture in Medical Software: Promises and Perils. J Am Med Inform Assoc. 2007 Mar-Apr; 14(2): 244–246.

[3] Sáez C, Martínez-Miranda J, Robles M, García-Gómez JM. Organizing quality assessment of shifting biomedical data. Stud Health Technol Inform. 2012;180:721-5.

[4] [Sáez C, Robles M, García-Gómez JM. Comparative study of probability distribution distances to define a metric for the stability of multi-source biomedical research data. In: Engineering in Medicine and Biol-ogy Society (EMBC), 2013 35th Annual International Conference of the IEEE; 2013. p. 3226–3229] .

[5] Erl, Thomas. *SOA Principles of Service Design*. Edición: 1. Prentice Hall, 2008.

[6] Erl, Thomas. *Service-Oriented Architecture: Concepts, Technology, and Design*. Edición: 1. Prentice Hall, 2005.

[7] Gamma, Erich. *Patrones de diseño*. Edición: 1. Madrid etc.: ADDISON WESLEY, s. f.

[8] Freeman, Eric, Elisabeth Robson, Bert Bates, y Kathy Sierra. *Head First Design Patterns*. Edición: 1. Sebastopol, CA: O'Reilly Media, 2004.

Tabla de figuras

Anexo 1. Esquema XML solicitud WS

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
 <xs:element name="dqv_request">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="params"/>
    <xs:element ref="data"/>
    <xs:element ref="meta_data"/>
    <xs:element ref="meta_inf"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="params">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="dimension"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="dimension">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="axis"/>
   </xs:sequence>
   <xs:attribute name="name" use="required" type="xs:NCName"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="axis" type="xs:NCName"/>
 <xs:element name="data">
  <xs:complexType mixed="true">
   <xs:attribute name="colums" use="required" type="xs:integer"/>
   <xs:attribute name="delim" use="required"/>
   <xs:attribute name="rows" use="required" type="xs:integer"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="meta_data">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="types"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="types">
  <xs:complexType>
   <xs:sequence>
    <xs:element maxOccurs="unbounded" ref="t"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="t" type="xs:NCName"/>
 <xs:element name="meta_inf">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="date"/>
   </xs:sequence>
   <xs:attribute name="generatedby" use="required" type="xs:NCName"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="date" type="xs:string"/>
</xs:schema>
```

Anexo 2. Esquema XML de resultados

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="myXmlResult">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="dimension"/>
        <xs:element ref="axis"/>
        <xs:element ref="rows_dataset"/>
        <xs:element ref="columns_dataset"/>
        <xs:element ref="values"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="dimension" type="xs:NCName"/>
  <xs:element name="axis" type="xs:NCName"/>
  <xs:element name="rows_dataset" type="xs:integer"/>
  <xs:element name="columns_dataset" type="xs:integer"/>
  <xs:element name="values" type="xs:string"/>
</xs:schema>
```