Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de servicios para una aplicación web colaborativa en el marco de la plataforma FIWARE

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Alberto García Simón

Tutor: Óscar pastor López

Co-Tutor: Francisco Valverde Giromé

Curso académico 2015-2016

Resumen

En la actualidad las plataformas Cloud (como AWS o Microsoft Azure) se han convertido en una solución tecnológica muy extendida a nivel empresarial, a la hora de desarrollar servicios (componentes software que ofrecen una funcionalidad que aporta un claro valor de negocio) que requieren de una infraestructura de alto rendimiento. Esta aproximación es especialmente interesante para las PYMEs tecnológicas que no disponen los recursos necesarios para gestionar de forma adecuada infraestructura de alto rendimiento.

Este proyecto aborda un caso de estudio real para desarrollar un conjunto de servicios en el contexto de una aplicación web colaborativa: GEMDomus. Dicha aplicación permite la elaboración de un diagnóstico médico mediante la integración de un gran volumen de información genómica proveniente de fuentes de datos abiertas. Dicha información es analizada mediante un proceso cognitivo que involucra a varios expertos que interactúan en una misma sala a través de distintos dispositivos (tableta, pc, pantalla proyectada, etc.). El objetivo de los servicios a desarrollar es procesar la información relevante, capturar la interacción realizada por los distintos usuarios simultáneamente y gestionar como la información es visualizada en cada dispositivo utilizando tecnologías Web.

Por el volumen de la información procesada y la complejidad del proceso de análisis, esta aplicación requiere de su despliegue en una plataforma cloud. En concreto, se utilizará FIWARE (www.fiware.org), un ecosistema abierto impulsado por la comisión europea para el desarrollo de aplicaciones inteligentes en múltiples dominios. A diferencia de las soluciones cloud propietarias, FIWARE aboga por el uso de interfaces y componentes software de código abierto, aspecto que influye positivamente en términos de interoperabilidad y costes. El presente proyecto analizará las tecnologías ofrecidas por esta plataforma para mejorar el desarrollo del caso de estudio propuesto. Por lo tanto, la contribución de este proyecto tiene dos vertientes: por un lado, se realizará un desarrollo de servicios cloud en un entorno industrial y, por otro lado, se evaluarán los potenciales beneficios de usar la plataforma FIWARE con dicho fin.

Palabras clave: Java, Spring, FIWARE, colaborativo, Websocket

Resum

Actualment les plataformes cloud (com ara AWS o Microsoft Azure) son una solució tecnològica molt emprada a nivell empresarial, per a desenvolupar serveis una infraestructura d'alt rendiment. Aquesta aproximació és especialment interessant per les PIMEs tecnològiques que no gaudeixen dels recursos necessaris per a gestionar adequadament aquestes instal·lacions.

Aquest projecte aborda un cas d'estudi real per desenvolupar un conjunt de serveis en el context d'una aplicació web col·laborativa: GEM Domus. L'esmentada aplicació permet l'elaboració d'un diagnòstic mèdic mitjançant la integració d'un gran volum de d'informació genòmica provinent de fonts obertes de dades. Aquesta informació es analitzada per mitja d'un procés cognitiu que involucra a diversos experts que interactuen en la mateixa sala amb distints dispositius (tablet, pc, pantalla de projecció, etc.). La fi dels serveis a desenvolupar es processar la informació rellevant, capturar la interacció simultània realitzada per els diversos usuaris i gestionar com la informació es visualitza en cadascun dels dispositius emprant tecnologies Web.

A causa del volum de la informació processada i la complexitat del procés d'anàlisis, aquesta aplicació requereix d'un desplegament en una plataforma cloud. En concret, s'utilitzarà FIWARE (www.fiware.org), un ecosistema obert impulsat per la comissió europea per a el desenvolupament d'aplicacions intel·ligents en múltiples dominis. A diferència de les solucions cloud propietàries, FIWARE advoca per l'ús d'interfícies i components de programari de codi obert, aspecte que influeix positivament en termes d'interoperabilitat i despeses. El present projecte estudiarà les tecnologies oferides per aquesta plataforma per a millorar el desenvolupament del cas d'estudi proposat. Aleshores, la contribució té dues vessants: per una banda, es desenvoluparà un conjunt de servicis tecnològics en un entorn industrial i, per l'altra, s'avaluaran els potencial beneficis d'usar la plataforma FIWARE amb aquesta finalitat. Aquest projecte analitzarà les tecnologies oferides per aquesta plataforma per a millorar el desenrotllament del cas d'estudi proposat. Per tant, la contribució d'aquest projecte te dues vessants: Per un costat, es realitzarà un desenrotllament de servicis cloud en un entorn industrial I, per altra banda, s'avaluaran els potencials beneficis d'utilitzar la plataforma FIWARE amb eixe fi.

Paraules clau: Java, Spring, FIWARE, col·laboratiu, Websocket

Abstract

Nowadays cloud platforms (AWS or Microsoft Azure) have become a widespread technology solution in the business world when developing services (software components that offer a functionality which provides a clear business value) that require a high performance infrastructure.

This project addresses a real case study to develop a set of services in the context of a collaborative web application: GEMDomus. This application allows the development of a medical diagnostic through the integration of a huge volume of genomic information from open data sources. This information is analyzed through a cognitive process involving several experts that interact in the same room through different gadgets (tablet, pc, projector, etc.). The aim of the services to develop is process relevant information, capture the interaction performed simultaneously by the different users and manage how the information is visualized on each dispositive using web technologies.

Because of the volume of the processed information and the complexity of the analysis process, this application needs to be deployed in a cloud platform. In particular, Fiware will be used (www.fiware.org), an open ecosystem driven by the European commission for intelligent application development in multiple domains. Unlike proprietary cloud applications, FIWARE advocates the use of open interfaces and components, aspect that positively influences on terms of interoperability and costs. This project will analyze the offered technologies by this platform to improve the development of the proposed case study. Therefore, the project contribution has two sides: On the one hand, development of cloud services in an industrial environment will be made. On the other hand, potential benefits of using FIWARE platform will be evaluated.

Keywords: Java, Spring, FIWARE, collaborative, Websocket

Tabla de contenidos

Índice de figuras

1. Introducción

1.1. Motivación

El ser humano es un ser social por naturaleza que puede adaptarse tanto a entornos colaborativos o como individualistas. Es una realidad que el trabajo colaborativo se ha abierto paso en el mundo informático en las últimas décadas con ejemplos como GNU/Linux o Wikipedia [1]. En palabras de Yochai Benkler[1], el trabajo colaborativo puede ser descrito como "el sistema de producción, distribución y consumo de bienes de información que se caracteriza por acciones individualizadas descentralizadas, ejecutadas a través de medios ampliamente distribuidos y ajenos al mercado y a sus estrategias" [2] .

El trabajo colaborativo proporcionan una serie de ventajas ya que se establece un compromiso y una interdependencia positiva entre los miembros del grupo, pues cada uno se siente responsable no solo de su propio trabajo, sino del de los demás, además de aumentar el sentido crítico ya que cada integrante del grupo se ve obligado a contrastar su interpretación con la del resto de integrantes del grupo [3] . Producto de esta filosofía de trabajo, se crean centros de trabajo colaborativos, caracterizados por ser espacios interconectados en los que todos los participantes acceden e interactúan entre ellos como una sola entidad [4] . Dicho espacio puede estar respaldado por elementos tecnológicos que les permitan compartir un modelo o información común; un ejemplo de tecnología al servicio de esta filosofía de trabajo sería el software colaborativo. Se entiende por software colaborativo a aquel tipo de software encargado de integrar todo el trabajo interdependiente que se encuentra en diversas estaciones de trabajo con muchos usuarios concurrentes conectados a través de una red en un solo proyecto [5] . Este tipo de software puede dividirse en herramientas de colaboración-comunicación, de conferencia o de gestión colaborativa o en grupo; esta última categoría es la más interesante pues se

[1] Catedrático de derecho empresarial en la Facultad de derecho de la universidad de Harvard, autor de "la riqueza de las redes" entre otros.

centra en facilitar y agilizar las actividades en grupo ayudando en la comunicación entre usuarios y la gestión del conocimiento.

Con el paso de los años las aplicaciones han evolucionado hasta convertirse en complejos sistemas que pueden requerir una gran capacidad de cómputo, de uso de datos o de concurrencia. Esto ha propuesto un nuevo reto que ha visto como la aparición de la tecnología *cloud* ha abordado todos estos nuevos inconvenientes proporcionando un nuevo paradigma que permite una gran escalabilidad, elasticidad y rendimiento. Gracias a esta tecnología, podemos reducir costes, pues ofrece un servicio de pago por uso sin necesidad de poseer Hardware, pudiendo ser su obtención inmediata y automatizada. Esto significa, también, que, ante picos de demanda, los recursos se adecúan a las necesidades dotando a la aplicación de una gran escalabilidad y elasticidad. Como se puede apreciar, el uso de plataformas *cloud* para el despliegue de aplicaciones es más que recomendable.

Desde la explosión del *Big data*, se ha visto cómo la cantidad de datos genómicos disponibles ha crecido exponencialmente. Gracias a las nuevas tecnologías, somos capaces de secuenciar y analizar cada vez más material genético y con ello intentar mejorar el diagnóstico de enfermedades relacionadas. No obstante, este aumento de datos ha venido acompañado de una mayor heterogeneidad de los datos; tenemos cada vez más datos, pero éstos están aislados, no se aprovecha todo su potencial, limitando así la labor investigadora y de diagnosis de los médicos y genetistas.

La precisión de estos análisis clínicos depende de dos factores:

- La cantidad y calidad de los datos: La heterogeneidad hace que, si bien se tenga acceso a una gran cantidad de repositorios de datos abiertos (*open data*), no se puedan comparar o utilizar conjuntamente. Se hace necesario un método que permita unificar la ontología[2] de manera clara y precisa, generando un modelo que incluya todo el conocimiento obtenido hasta el momento.
- La facilidad de diagnóstico: Debido a la complejidad de los datos, es de vital importancia proporcionar un entorno colaborativo y altamente participativo donde el equipo de expertos pueda aprovechar todo su conocimiento y

[2] Se entiende por ontología una definición formal de tipos, propiedades y sus relaciones. En este caso, para el ámbito bioinformático; especialmente terminología relativa a conceptos genéticos.

experiencia a partir de los datos disponibles. La simplificación en la visualización y filtrado de los datos es vital.

Supongamos, por ejemplo, que un centro de investigación genómica español necesita ciertos datos para llevar a cabo su investigación sobre la enfermedad hereditaria X. Estos datos se componen de un conjunto de 400 exomas[3] de pacientes tanto sanos como pacientes con la enfermedad con el objetivo de tener un conjunto de datos representativo y completo. A pesar de que el coste de la secuenciación de ADN se está reduciendo, sigue siendo dificultoso y conlleva una gran cantidad de tiempo (Cada exoma son 50 GB) encontrar y clarificar las muestras para una enfermedad específica. En el caso de enfermedades extrañas resulta todavía más difícil encontrar potenciales candidatos. Si, por ejemplo, existe otro centro de investigación en Francia que posee los datos necesarios para la investigación española, estos no podrían ser enviados directamente pues probablemente estarán almacenados siguiendo un modelo diferente, sin preocuparse de haber implementado algún mecanismo de interoperabilidad. Además, hay diferencias semánticas potenciales en cómo se ha almacenado los datos (Información en francés, por ejemplo). Esto unido al gran tamaño de los datos, hace imposible enviar los datos sin ningún tipo de preprocesamiento.

Con este sencillo ejemplo se puede observar cómo los factores anteriormente nombrados influyen claramente en el diagnóstico. Es más que deseable una herramienta que permita solventar o, al menos, paliar estos problemas. Se plantea el emocionante reto de conseguir desarrollar una aplicación *cloud* con el firme propósito de, aprovechando las ventajas del trabajo colaborativo y las técnicas de programación e interacción persona-computador, sea capaz de facilitar la ardua labor de diagnóstico a los profesionales de la salud. Este TFG se ha centrado en el desarrollo de la lógica de negocio de la aplicación; para ello, en primer lugar, se ha desarrollado un conjunto de servicios de recuperación y transformación de datos permitiendo su presentación al usuario en distintos formatos (descritos en las secciones cinco y seis) modificables en tiempo de ejecución. Esto permite una mayor flexibilidad y rapidez, lo cual redunda en una mejora del diagnóstico. En segundo lugar, la implementación de un sistema colaborativo que permita una interacción en tiempo real de todos los usuarios donde ante cualquier cambio la actualización de todos los participantes sea inmediata.

[3] Parte del genoma formado por las partes codificantes de los genes que formarán parte del ARN mensajero maduro y darán lugar a las proteínas.

1.2. Objetivos

El trabajo tiene por objetivo principal el desarrollo de un servicio de recuperación y gestión de datos para una aplicación web en el ámbito de la bioinformática y analizar la idoneidad de la plataforma *cloud* donde se ha desplegado. Por un lado, motivado a partir del estudio de un caso de estudio real donde se plantea la problemática de la heterogeneidad de los datos y su interpretación; surge la necesidad de desarrollar un servicio de recuperación de datos que se adecúe a las necesidades concretas de esta área de trabajo. Con dicho fin se plantea la creación de un servicio con una arquitectura que cumpla los siguientes requisitos:

- Compatibilidad con un entorno de trabajo altamente colaborativo en el que interactúan simultáneamente varios usuarios.
- Soporte para la interacción en tiempo real con un frecuente flujo de intercambio de datos entre los clientes y el servidor
- Actualización instantánea (mecanismo push) de todos los dispositivos clientes ante un cambio.
- Capacidad de tratamiento de datos masivos.
- Alta escalabilidad.
- Simplicidad de uso.

Partiendo de las necesidades del caso de uso, el servicio debe ser capaz de proporcionar una lista con los estudios de secuenciación de ADN disponibles, así como las fuentes de datos disponibles para que el usuario pueda seleccionar los datos que desee analizar. Una vez seleccionados los datos, el servicio les dará el formato adecuado para que la interfaz de usuario pueda presentarlos utilizando gráficos sencillos que permitan al usuario filtrar la información por varios criterios, previamente definidos, ayudando y agilizando el proceso de diagnóstico. El proceso de selección y filtrado debe ser transparente para el usuario, permitiendo que se preocupe en el qué y no en el cómo, que es delegado en el servidor. Finalmente, debemos proporcionar una vista que contenga todas las variaciones que, partiendo de las seleccionadas inicialmente, cumplan los filtros que el usuario ha ido aplicando al conjunto de datos; dicha vista debe ser ordenable por varios criterios. Todos los elementos deben actualizarse de manera instantánea y transparente cuando se realice un cambio, ya sea ampliando el conjunto de datos o filtrándolo.

Por otro lado, el estudio y análisis de Fiware, una plataforma *cloud* impulsada por la Comisión Europea para el desarrollo y despliegue de aplicaciones de la Internet del Futuro utilizando interfaces y componentes de código abierto. Se ha analizado su arquitectura en detalle, las características y novedades que ofrece, así como la documentación disponible para su uso y las nuevas tecnologías que conforman esta plataforma [6], [7] para finalizar con una evaluación de los potenciales beneficios y problemas surgidos durante todo el proceso.

1.3. Estructura de la memoria

Este TFG se estructura en seis partes bien diferenciadas. La primera de ellas se centra en describir las tecnologías empleadas en el desarrollo y testeo de los servicios, así como las utilizadas por el resto de componentes de la aplicación web, analizando las ventajas que cada una de ellas ha proporcionado al desarrollo del proyecto. Se ha realizado un especial énfasis en las tecnologías empleadas para la comunicación e intercambio de información, eje principal de los servicios.

Seguidamente se ilustra el caso de uso que ha motivado el desarrollo de los servicios y su finalidad dentro de este, proporcionando un acercamiento pragmático a la aplicación web. Además, se desglosa el caso de uso analizando el problema que se plantea y sus características concretas, así como sus posibles soluciones y futuras complicaciones.

Posteriormente, se llega a uno de los ejes principales de este TFG, el análisis la plataforma *cloud* Fiware. Empezando por sus orígenes como proyecto FP7 de la Comisión Europea para llegar a un detallado análisis de su arquitectura, introduciendo cada uno de los componentes que lo forman; los componentes disponibles y los utilizados también han sido descritos y estudiados, con especial énfasis en su funcionalidad. Finalmente se han enumerado los problemas encontrados durante su utilización para la implementación tanto de los servicios como del conjunto de la aplicación web.

En el cuarto punto procedemos al análisis de la arquitectura de la aplicación. A pesar de que el TFG esté centrado en el desarrollo de los servicios de la capa de negocio, se incluye el modelo de la capa de persistencia y de la interfaz gráfica de usuario (*GUI*) para dotar a los servicios de una visión global dentro del conjunto de la aplicación web.

Debido a la importancia del intercambio de información, cada componente presenta su propia sección.

A continuación, llegamos al segundo de los ejes principales, la implementación de los servicios, detallando el procedimiento seguido y las soluciones empleadas para el intercambio de información y actualización automática de la *GUI* de todos los clientes. Se detalla la API con la que se interactúa con el servidor además de proporcionarse los detalles para comprender la implementación seguida como un todo que ofrece un servicio robusto y seguro donde los componentes del servidor, debidamente estructurados e intercomunicados, realizan correcta y eficientemente su funcionalidad.

Finalizamos con las conclusiones tras el desarrollo de los servicios y su posterior despliegue a la plataforma Fiware, enumerando las lecciones aprendidas tanto a nivel tecnológico como con la plataforma Fiware y el caso de estudio.

2. Tecnologías empleadas

2.1. Capa de persistencia

Los servicios desarrollados consumen una gran cantidad de datos. Estos datos han sido almacenados en POSTGRESQL, un sistema de gestión de bases de datos relacionales orientado a objetos y de código abierto. Ofrece una alta concurrencia gracias a un sistema llamado Acceso concurrente multiversión (MVCC) que permite acceder a una tabla mientras está siendo modificada donde cada usuario tiene acceso a la última versión consistente (*commit*) eliminando la necesidad de bloqueos explícitos [8] . En la sección Modelo se explicará detalladamente tanto el modelo lógico como físico

Además de la amplia variedad de tipos que ofrece, el usuario tiene la posibilidad de crear sus propios tipos indexables, posee disparadores (*triggers*) para definir acciones específicas de acuerdo a eventos y es posible definir bloques de código que se ejecutan en el servidor (en diversos lenguajes de programación) a modo de funciones [9] .

Con el objetivo de testear la funcionalidad del servicio desarrollado con independencia de la disponibilidad o carga de los datos legítimos, ha sido necesario el uso de una herramienta que permitiera poblar la base de datos con grandes cantidades de información desechables. Para ello se ha utilizado la herramienta PowerDesigner, desarrollada por SAP. Se trata de una herramienta de modelado que soporta *Model driven architecture* (MDA). Este paradigma permite que, a partir del modelo de la base de datos y definiendo los mapeos necesarios, se automatice la creación de grandes cantidades de datos tanto para testeo como para medir la eficiencia de las distintas aproximaciones implementadas en cuanto a tiempo de computación.

2.2. Capa de lógica de negocio

Para el diseño de la capa lógica de los servicios se ha optado por utilizar JAVA en su versión e*nterprise* como lenguaje de programación. JAVA es un lenguaje de programación orientado a objetos e independiente de la plataforma donde se ejecute. Se ha utilizado en su versión 1.8 debido a las mejoras introducidas en la gestión de colecciones, ofreciendo

soporte a operaciones propias del paradigma funcional. Específicamente, se incluye un nuevo tipo de colección que soporta operaciones de agregación de manera secuencial y paralela (Streams), la inclusión de expresiones lambda, y la interfaz Collector, que permite realizar operaciones de reducción mutables. Estas nuevas características son especialmente deseables en el tratamiento y filtrado de los datos [10] .

El intercambio de mensajes entre el servidor y los clientes es constante, motivo por el cual resulta de especial importancia el formato de estos. Para ello se ha elegido JSON, un formato ligero de intercambio de datos. Consta de dos tipos de estructuras: Una colección de pares nombre-valor y una lista ordenada de valores. Se trata de un formato independiente del lenguaje y fácilmente entendible [11] . Aprovechando la gran cantidad de librerías de que dispone dispone JAVA, se ha optado por utilizar Jackson. Jackson es una librería de JAVA para procesar mensajes en formato JSON. Se ha utilizado la versión core de la librería, que incluye el analizador sintáctico (*parser*), el generador de abstracciones y la implementación estándar para el manejador de tipos de JSON [12] . Jackson utiliza un modelo de objetos donde el objeto JSON completo es almacenado en memoria en un formato de árbol, el cual es navegado, analizado y modificado, en contraposición al modelo de flujo, donde los datos son leídos o escritos por bloques; este modelo consume más recursvos pero es muy flexible para manipular el contenido. Jackson está especialmente indicado para la conversión de datos de gran tamaño, dando unos excelentes resultados tanto en serialización como en deserialización [13] y puesto que los servicios desarrollados deben enfrentarse a una gran, y en crecimiento, cantidad de datos, era la opción óptima [14] . Se ha utilizado para serializar y deserializar los mensajes que se intercambian entre los clientes y el servidor en formato JSON.

```
{"example1":1, "example2":"2", "example3":["one","two"]}
```

Ilustración 1 ejemplo de JSON

Para interactuar con la base de datos es necesario el uso de un *framework* [4] que implemente mapeo objeto relacional. Utilizamos un *framework* que implemente el estándar JPA. Éste estándar consiste en una interfaz de programación de aplicaciones (API) de persistencia desarrollada para JAVA cuyo objetivo es no perder las ventajas de la orientación a objetos al interactuar con una base de datos. Esto lo consigue

[4] Se trata de una estructura tecnológica formada por artefactos o módulos que sirve de base para la organización y desarrollo de software

proporcionando un modelo de persistencia basado en clases simples (POJO) para el mapeo objeto-relacional donde cada entidad es persistida de manera asociada a una tabla de la base de datos que esté mapeando. Dicho mapeo se puede realizar utilizando anotaciones de JAVA o en un documento propio en formato XML. Si bien no implementa ningún tipo de mapeo objeto relacional (ORM) [15] ya que se trata de un documento presente en la especificación de las *Enterprise* JAVA *Beans*[5] que proporciona una serie de principios básicos para la gestión de la capa de persistencia en aplicaciones JAVA. Se utiliza como parte del *framework* Hibernate en la gestión de la capa de persistencia de la aplicación.

Como se ha mencionado, JPA tan solo es un estándar que ofrece pautas para gestionar la capa de persistencia [16]. En este proyecto se ha utilizado Hibernate como *framework* ORM, el cual implementa el estándar JPA. Facilita el desarrollo de aplicaciones permitiendo desarrollar clases persistentes que mapean tablas de la base de datos correspondiente. Permite inicializar los componentes bajo demanda (*lazy initialization)* y proporciona numerosas estrategias de unión entre tablas además de ser altamente configurable y escalable [17] .

El *framework* de Hibernate se divide en varios componentes:

- Persistente object: POJOs asociados a una sesión. Una vez la sesión sea cerrada podrán ser usados libremente por la aplicación, pasando a ser objetos no persistentes (*transient objects*).
- Configuración: El primer objeto que se crea en una aplicación Hibernate, almacena la configuración. Se crea durante la inicialización de la aplicación.
- SessionFactory: Se trata de del mapeo de una base de datos creado a partir del objeto Configuración. Se trata de un objeto inmutable y permite ser accedida por varios hilos de ejecución simultáneamente de manera segura (*thread safe*). Se utiliza para abrir y cerrar sesiones.
- Session: Proporciona una interfaz entre la aplicación y los datos almacenados en la base de datos. Encapsula la API de ejecución de operaciones sobre bases de datos desde JAVA (JDBC). Creado a partir de un objeto SessionFactory, actúa como una factoría de Transaction, Query y Criteria. Mantiene una caché de los objetos persistentes con los que trabajar.

[5] Enterprise JAVA Beans (EJB) es una API perteneciente al estándar de construcción de aplicaciones empresariales (*JEEK*) de Oracle Corporation

- Transaction: Realiza acciones atómicas y específicas sobre la base de datos. Provee de métodos para gestionar las transacciones.
- Query: Consulta utilizada sobre la base de datos. Puede estar en lenguaje SQL o HQL. HQL es el lenguaje de consulta utilizado por Hibernate, con notables similitudes a SQL, se diferencia de este en que es completamente orientado a objetos y comprende conceptos como herencia o polimorfismo.
- Criteria: Estos objetos se utilizan para crear y ejecutar sentencias orientadas a objetos y basadas en criterios

```
List example = session.CreateCriteria(Persona.class)
    .add(age.eq(new Integer(1)))
    .list();
```

Ilustración 2 Ejemplo de Criteria

Hibernate Tools es una suite de herramientas implementadas para Hibernate como un plugin de Eclipse. Forma parte del núcleo de JBoss Tools y permite de una forma automatizada mapear el modelo relacional de la base de datos a un conjunto de POJOs que actuarán como objetos persistentes y enlazarán la capa lógica con la de persistencia [18] .

Debido a la naturaleza de la aplicación, se requiere un intercambio continuo de mensajes así como la actualización instantánea ante cambios. Por ello, se requiere un protocolo de comunicación completamente diferente a los utilizados tradicionalmente en la red. En origen, internet fue creado bajo un protocolo de petición-respuesta en un modelo cliente/servidor llamado *Hypertext Transfer Protocol* (HTTP). De este modo, el servidor se limitaba a responder peticiones de los clientes y por cada petición se realizaba una conexión diferente. Con el crecimiento de la web y el aumento de la interactividad se debió redefinir la conectividad para reducir la latencia. Si bien a partir de HTTP/1.1 se introducen conexiones reutilizables, estas seguían siendo redundantes ya que el protocolo HTTP es *stateless* (no almacena información sobre su estado) por lo que, a pesar de reutilizar la conexión, se envía información redundante. Dada su naturaleza, HTTP posee un modo de envío donde la información es bidireccional pero no simultáneo (*half duplex*), lo que convierte el protocolo en ineficiente en ambientes interactivos o colaborativos.

Se ha intentado proporcionar tecnologías que hagan posible el desarrollo de aplicaciones que interactúen en tiempo real mediante la creación de nuevas tecnologías

(*Polling, long Polling*) pero estas tienen sus limitaciones: *Polling* se basa en realizar peticiones al servidor con un intervalo de tiempo, tras la petición la conexión se cierra y en cada petición debe abrirse una nueva; *Long polling* sigue el mismo método que *Polling* pero manteniendo abierta la conexión durante un tiempo predeterminado, teniendo que reconectarse periódicamente [19] .

Como solución, los Websockets proporcionan una conexión *full duplex* donde, a partir de una única petición, se crea una conexión bidireccional simultánea que permanecerá abierta hasta que el cliente o el servidor la cierre. Se trata de un protocolo creado en 2011 que proporciona una API desarrollada por el *World Wide Web Consortium* (W3C). Dicha API permite abrir y cerrar conexiones, enviar y recibir mensajes y la creación de *listeners* que, ante eventos lanzados por el servidor, ejecuten una acción [20]. Esta tecnología está soportada por la mayoría de navegadores (Tabla 1 *Navegadores con soporte para WebSockets*) y permite que la comunicación en tiempo real sea mucho más eficiente al estar basada en un único protocolo de mensajes.

Navegador	**Versión compatible**
Internet Explorer/ Edge	11
Firefox	45
Chrome	29
Safari	9
Opera	36
IOS Safari	8.4
Android browser	4.4
Chrome for Android	49

Tabla 1 Navegadores con soporte para WebSockets

Para establecer una conexión mediante WebSockets, el cliente (navegador compatible con tecnología WebSocket) inicia el proceso. Este consiste en un *handshake*[6] consistente en el envío de una petición GET cuya versión de HTTP sea igual o mayor a la 1.1 en la cual se solicita al servidor una mejora o *upgrade* de la conexión. Si los parámetros de la cabecera son correctos y la petición es procesada correctamente, el servidor responderá vía HTTP informando de que la actualización de la conexión se realizó correctamente mediante el código 101 [21] . Este código confirma el correcto cambio de protocolo de comunicación para dicha conexión estableciendo una conexión *full dúplex* [22].

[6] Se define como un proceso automatizado de negociación donde, de manera dinámica, se establecen los parámetros de un canal de comunicación establecido entre dos entidades.

Se ha utilizado este componente para establecer una comunicación en tiempo real entre el servidor y los clientes que permita una actualización instantánea y altamente escalable de los datos utilizados cuando estos sean modificados o filtrados. Los clientes, mediante el uso de la API que proporciona el servicio, son capaces de conectarse al servidor y obtener diversas colecciones de datos debidamente preprocesadas para su visualización además de poder modificar dicha visualización mediante la adición o eliminación de filtros. El servidor es el encargado de recibir dichas modificaciones y comunicarlas al resto de clientes.

Finalmente, se ha empleado Spring como *framework* para construir la aplicación, debido a las ventajas que ofrece su uso; entre muchas otras, cabe destacar las siguientes: Spring proporciona inyección de dependencias. Esta técnica permite crear instancias de objetos e inyectarlas en tiempo de ejecución. Se basa en dos conceptos clave: JAVABeans e interfaces. Al usar Spring como proveedor de inyección de dependencias se gana flexibilidad ya que se puede definir dicha configuración en diferentes formas, ya sea con un archivo XML, anotaciones en las clases, etc. El uso de interfaces permite a Spring el uso de *dynamic proxies*. Estos son clases que implementan una lista de interfaces en tiempo de ejecución cuando son creados. Además, el módulo de acceso de datos de Spring proporciona soporte para Hibernate e implementando una API más sencilla [23] .

A partir de la versión 4.0 Spring proporciona soporte para el uso de Websockets facilitando su propia implementación del estándar (JSR-3356). Ofrece una serie de interfaces a implementar permitiendo acceso de bajo nivel a la API pero manteniendo la simplicidad. Se ha optado por utilizar Spring Boot ya que añade automáticamente todas las dependencias necesarias para el proyecto en el momento de su creación y, además, elimina la necesidad de configurar Spring utilizando XML. Spring Boot permite configurar mediante el uso de anotaciones en el código JAVA la mayor parte de la configuración de las aplicaciones web, simplificando el proceso en gran medida [24].

2.3. Gestión del proyecto

Además de las tecnologías empleadas en el desarrollo como tal de la aplicación (lenguajes de programación, *frameworks*, etc.) diversas tecnologías han sido utilizadas para el despliegue de la aplicación y para el manejo de dependencias externas.

Maven es un *framework* de administración de dependencias perteneciente a Apache que simplifica la creación, testeo y empaquetamiento de proyectos [25] . Proporciona un estándar de cómo el código de la aplicación debe ser estructurado facilitando la portabilidad de proyectos entre ambientes de desarrollo (IDE). Permite declarar las dependencias externas de un proyecto en un documento independiente escrito en formato XML siguiendo un formato predeterminado y, a partir de este documento llamado pom.xml, se descargan y empaquetan automáticamente siguiendo un paradigma declarativo ya que se le indica qué dependencia se desea y Maven se encarga de cómo importarla [24] . Maven proporciona arquetipos (*Archetypes*) como plantillas de modo que al crear un proyecto este se inicia con el árbol de directorios y las dependencias listas. Además de la administración de dependencias, se ha utilizado para empaquetar y construir la aplicación en formato war para ser desplegado en el servidor Apache Tomcat [25] .

Con el fin de enviar información de manera segura entre servidor y cliente, se ha utilizado el protocolo *Secure Socket Layer* (*SSL*). Un certificado *SSL* es un pequeño archivo que posee una clave criptográfica firmada digitalmente con los detalles de la organización. Cuando se utiliza en un servidor, activa el protocolo HTTPS sobre el puerto 443. Existen diversos tipos de certificados *SSL* [28] en función del nivel de seguridad requerido. Debido a los requerimientos de seguridad de FIWARE (Ver sección 4), ha sido necesario implementar una comunicación segura mediante https entre el servidor y FIWARE. Para ello, se ha utilizado el software OpenSSL, que permite crear certificados de seguridad autofirmados que son añadidos a un almacén de claves (*Keystore*) mediante la herramienta JAVA Keytool. Dicho almacén de claves lo utiliza el servidor para servir conexiones seguras e identificarse como un servidor confiable [29] .

Una vez finalizados los servicios y empaquetados gracias a Maven, estos deben ser accesibles desde la red. Para ello se ha utilizado Apache Tomcat. Se trata de un contenedor de *servlets* (clase JAVA utilizada para ampliar las capacidades de un servidor), es decir, funciona como un servidor de aplicaciones JAVA. A pesar de venir con licencia *open-source*[7], posee todas las características de un contenedor de aplicaciones web comercial, proveyendo además funcionalidad extra como su gestor de aplicaciones. Tan solo se puede levantar una instancia de Tomcat por cada máquina virtual de JAVA (JVM)

[7] Estas licencias permiten que tanto el código fuente como los archivos binarios puedan ser modificados y redistribuidos libremente siempre y cuando se distribuya bajo la misma licencia.

de modo que si una falla, las demás pueden seguir funcionando. Una instancia de Tomcat consiste en un grupo de contenedores de aplicaciones con una jerarquía claramente definida [30] :

```
<Server>
        <Service>
                <Connector />
                <Engine>
                        <Host>
                                <Context> </Context>
                        </Host>
                </Engine>
        </Service>
</Server>
```

Ilustración 3 Jerarquía de componentes de Tomcat

- Server: El primer contendor, representa el motor completo de una instancia Tomcat y puede contener uno o más contenedores de servicios.
- Service: Contiene una colección de conectores que comunican con un único Motor (Engine)
- Connector: define la clase que realiza el manejo de peticiones y respuestas a partir de las llamadas de los clientes a la aplicación.
- Engine: Cada servicio posee un único Engine, el cual administra todas las peticiones que recibe a través de los conectores.
- Host: Este elemento define los hosts virtuales contenidos en el Engine.
- Context: Representa una aplicación web que se está ejecutando en el Host.

3. Caso de estudio

Desde la creación del proyecto del genoma humano, las técnicas de secuenciación de ADN han evolucionado dando lugar a una auténtica revolución que ha dado lugar a la medicina personalizada, donde cada individuo puede ser genéticamente analizado, facilitando así el diagnóstico de enfermedades hereditarias o de causa genética. Propiciado por el gran descenso del coste de la realización de los estudios genéticos (en el año 2000 el coste de secuenciación de un genoma era de diez millones de dólares, actualmente el coste aproximado es de aproximadamente mil dólares) así como por la mejora de las tecnologías empleadas para obtener muestras de ADN y su secuenciación (especialmente utilizando tecnología de secuenciación de nanoporos [31] o mediante la utilización de fluoróforo).

A pesar de estos avances, los análisis genéticos que se realizan para el diagnóstico de enfermedades son diagnósticos dirigidos, es decir, tan sólo se utilizan para complementar un estudio realizado previamente. Esto es debido a que el estudio genético se inicia a petición de un médico y en función de una sintomatología o historial clínico que indiquen que puede existir una enfermedad de origen o causa genética. El resultado de dicho análisis genético consiste en un informe de soporte al diagnóstico donde se detallan las anomalías que han sido detectadas en el genoma del paciente que estén relacionadas con enfermedades para poder concluir si dicho paciente es genéticamente potencialmente vulnerable o no a una enfermedad. Estas variaciones se pueden comparar con la información existente en repositorios públicos para una mayor fiabilidad [32] .

Debido a la mayor viabilidad de la realización de los análisis genéticos, se ha producido una explosión de datos con un ingente número de análisis realizados y variaciones detectadas. Existe un gran conjunto de repositorios que contiene información genómica y cada uno de ellos tiene su propio criterio de estructuración, por lo que la información es representada de manera diferente, sin que existan estándares para ello. Este hecho causa una gran dispersión y heterogeneidad que redunda en una baja eficiencia de los análisis genéticos, pues requiere conocer cómo y dónde está almacenada la información deseada y recolectarla de manera natural [33] .

En conclusión, nos encontramos ante una gran cantidad de datos que presentan un alto grado de heterogeneidad donde el componente tecnológico y su complejidad es cada vez mayor, pero los profesionales carecen de los conocimientos informáticos necesarios para poder ordenar y procesar de manera eficiente el conjunto de datos y herramientas disponibles. Es necesario el análisis de las posibles soluciones disponibles a través de la adopción de estándares (asumiendo un único modelo de datos común) y el desarrollo de aplicaciones que permitan, por una parte, superar el hándicap del desconocimiento tecnológico permitiendo interactuar de manera simple y efectiva a los profesionales del sector; y por otra parte, aumentar la eficiencia de los análisis genéticos así como asegurar que los nuevos datos generados a partir de estos permanecerán ordenados y de fácil y rápido acceso.

El propósito de este TFG es intentar ayudar a superar estas barreras técnicas y conceptuales a través del desarrollo de unos servicios que, una vez integrados en la aplicación, superen la complejidad de la variedad de los datos genómicos e incluya el conocimiento clínico para mejorar su fiabilidad.

4. Análisis de la plataforma Fiware

Acto seguido, se llega a uno de los ejes principales de este TFG, el análisis la plataforma *cloud* Fiware. Empezando por sus orígenes como proyecto FP7 de la Comisión Europea para llegar a un detallado análisis de su arquitectura, introduciendo cada uno de los componentes que lo forman; los componentes disponibles y los utilizados también han sido descritos y estudiados, con especial énfasis en su funcionalidad. Finalmente se han enumerado los problemas encontrados durante su utilización para la implementación tanto de los servicios como del conjunto de la aplicación web.

4.1. Análisis tecnológico

Fiware es una nueva infraestructura surgida como proyecto FP7 de la Comisión Europea cuando en 2011 la Comisión Europea y las principales empresas TIC europeas emprendieron un programa de colaboración público-privada (PPP) y cuyo fin es la creación y despliegue de servicios y aplicaciones en internet; representando una opción abierta para el desarrollo y despliegue global de aplicaciones en la internet del futuro. Esta nueva infraestructura está ubicada en la nube para cumplir su objetivo, proporciona un conjunto de APIs abiertas y completamente libres de *royalties* (pago que se efectúa al titular de dederechos de autor o patente a cambio del uso o explotación de este) para el desarrollo rápido de aplicaciones en numerosos sectores facilitando así la reutilización e introduciendo nuevos estándares [34] . Separa la cadena de valor de las aplicaciones de forma que la plataforma, el desarrollo de los servicios o su despliegue (entre otros) puedan ser proporcionados por entidades diferentes, ayudando de esta formar a su expansión con el claro objetivo de mejorar la competitividad [6] . Fiware está basado en un conjunto de elementos llamados *Generic Enablers* (serán explicados detalladamente en la sección 4.1) que actúan esencialmente como programas reutilizables. Estos componentes están disponibles en el catálogo de Fiware.

El objetivo de Fiware no solo es ofrecer alternativas tecnológicas sino crear un ecosistema que proporcione mejores oportunidades a sus posibles clientes. Este ecosistema se cimienta en cinco pilares básicos [35] :

- Plataforma Fiware: Porporciona un conjunto de APIs estandarizadas tan simples como potentes para facilitar el desarrollo de futuras aplicaciones que hagan uso de internet. Este conjunto de APIs son públicas y gratuitas. Esta plataforma proporciona, por una parte, almacenamiento en la nube basado en el estándar OpenStack[8] mejorado y, por otra parte, una librería de componentes ofrecidos como servicios. Esta librería de componentes proporciona un conjunto de APIs estandarizadas con el objetivo de facilitar en gran medida el desarrollo de futuras aplicaciones basadas en internet (por ejemplo, proporciona componentes que permitan el procesamiento en tiempo real de grandes cantidades de datos o la incorporación de interfaces web de usuarios avanzadas). Estos componentes se conocen como *Generic Enablers* (GE) y sus especificaciones son públicas y gratuitas, permitiendo la existencia de diversas implementaciones para cada uno de los GE. Si bien pueden existir diversas implementaciones de un mismo GE (GEi), tan sólo existe un GE de referencia (GEri).
- Fiware Lab: Instancia de trabajo de la plataforma Fiware creada para realizar pruebas libremente. En este entorno de pruebas los desarrolladores pueden mostrar al mundo sus ideas a potenciales inversores y clientes permitiendo, a su vez, un lugar donde inversores y clientes busquen y encuentren más fácilmente lo que buscan. Se trata de un punto de encuentro para la innovación y creación.
- La suite Fiware Ops tools: Es un conjunto de herramientas que facilitan el despliegue, configuración y uso de las instancias de Fiware por los proveedores. Está diseñada para ayudar a expandir la infraestructura asociada a una instancia de Fiware añadiendo los nodos necesarios cuando sea necesario así como permitir la cooperación de múltiples proceedores. En resumen, esta suite es la herramienta utilizada para construir, operar y expandir Fiware Lab.
- Fiware Accelerator programme: Proporciona un paraguas que da soporte a programas específicos cuyo objetivo sea ayudar a movilizar recursos que ayuden a emprendedores a desarrollar ideas innovadoras utilizando Fiware. El primer programa lanzado por la comisión europea ha movilizado cien millones de euros delos cuales el 80% será dado a pequeñas y medianas empresas (SMEs) y startups que utilicen Fiware. Su objetivo es transformar las mejores ideas en los negocios

[8] Se trata de un proyecto de computación en la nube siguiendo el paradigma de infraestructura como servicio (IaaS) de código abierto y distribuido najo licencia Apache. Para más información visitar: www.openstack.org

exitosos que tanto se demandan en la actualidad en el mercado digital mediante la financiación, asesoramiento, formación y creación de redes.

- Fiware mundus programme: Si bien Fiware ha sido creado en Europa, está diseñado con una intención global, con intención de expandirse a otras regiones. Algunos países de América Latina como México, Brasil o Chile ya han decidido unirse a Fiware y trabajan en la creación de nodos Fiware.

Con la creación de este ecosistema, Fiware pretende contribuir al crecimiento económico de las regiones donde sea adoptado. Por ello, y para ayudar a los emprendedores que decidan hacer uso de Fiware, se ha creado Fiware Academy[9]; un lugar donde encontrar cursos de aprendizaje, lecciones y contenido de diversa índole. Los cursos están agrupado por categorías. En Mayo de 2016, posee un total de 78 cursos divididos en 3 categorías: Fiware Enablers, Fiware Ops y Fi-STAR (e-health) [36] .

Una vez introducidos los pilares del ecosistema de Fiware, resulta conveniente definirlos de una manera más precisa y detallada. Siguiendo el anterior orden de cita, empezaremos con la plataforma Fiware, para continuar con Fiware Lab, prosiguiendo con Fiware Ops tools y finalizando con los programas mundus y Accelerator.

El primer elemento a analizar es la plataforma Fiware, con especial énfasis en la arquitectura de referencia utilizada. Esta plataforma ha tenido un gran crecimiento y en 2016 posee 6671 usuarios. En Mayo de 2016, existen doce regiones entre las cuales se reparten 2928 núcleos, 10438 GB de memoria ram, 577 TB de memoria persistente, 5394 ips y un total de 1333 máquinas virtuales instanciadas (Información sujeta a continua actualización); la gran mayoría de estos recursos están destinados a la región española [37] .

Región	Núcleos (físicos)	Memoria RAM en GB (física)	HDD en TB	IPs	Máquinas virtuales
Sao Paulo (Brasil)	48	153,4	0,8	8	1
España	1376	5602	430	4079	708
Lannion (Francia)	176	345,5	9,3	120	153
Sophia Antipolis (Francia)	128	125,7	2	55	28
Volos (Grecia)	192	566,7	17,9	54	7

[9] Disponible en: http://edu.fiware.org/

Pireo (Grecia)	112	219,9	1,7	104	16
Creta (Grecia)	-	-	-	-	-
Trento (Italia)	192	377,3	4,7	164	75
Poznan (Polonia)	120	152,4	29,5	71	64
Praga (República Checa)	192	1000	11,6	30	113
Zúrich (Suiza)	192	1511,2	17,7	211	47
Budapest (Hungría)	200	383,9	51,8	498	121
12 Regiones	**2928**	**10438**	**577**	**5394**	**1333**

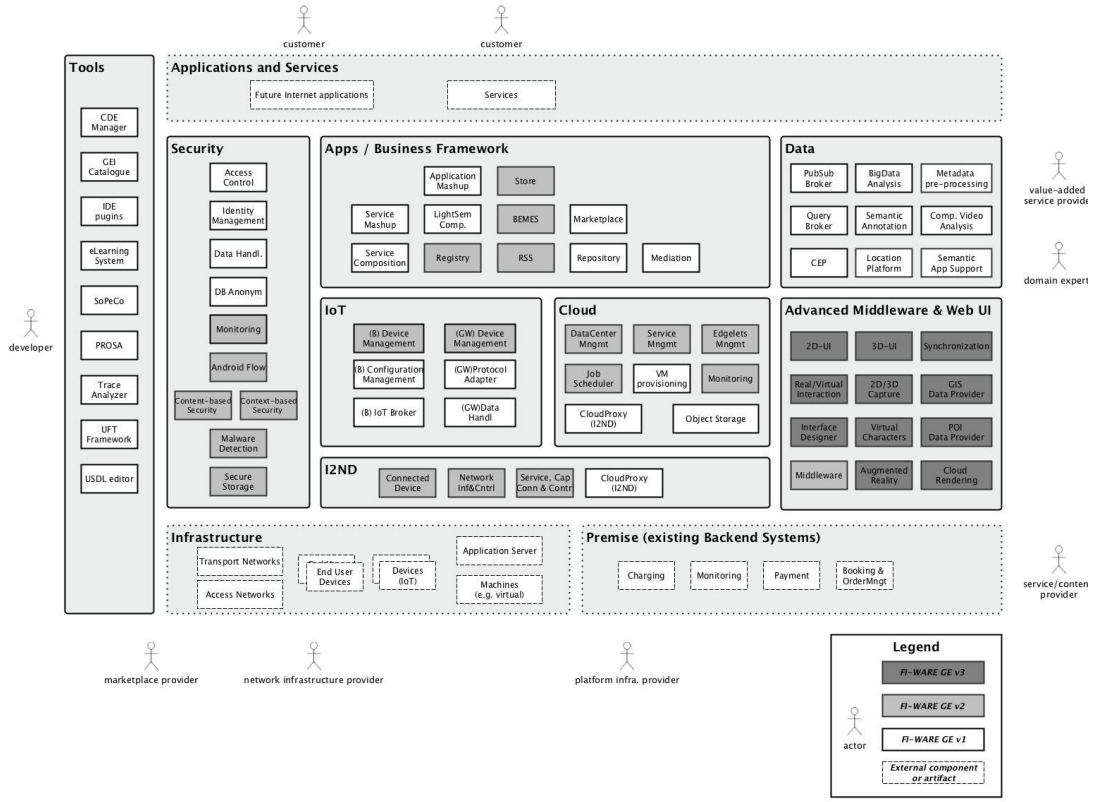


Ilustración 4 Esquema de la plataforma Fiware

Esta arquitectura de referencia se puede instanciar en una arquitectura concreta seleccionando e integrando los GE correspondientes, pudiendo ser modificada mediante la eliminación o agregación de otros GE.

Antes de entrar a definir la arquitectura, resulta conveniente enumerar los roles existentes en Fiware.

Rol	Descripción
Desarrollador de aplicaciones	Futuros desarrolladores de aplicaciones, se les estimula a desarrollar aplicaciones inteligentes centrándose ya sea por los grandes mercados o en pequeñas empresas y organizaciones. Estas aplicaciones deberían ofrecer flexibilidad en su despliegue y ejecución en la nube. Estas aplicaciones están destinadas a implementar un conjunto de funcionalidades para ser exportadas como un servicio a los usuarios finales mediante un conjunto de interfaces y APIs bien definidas. Normalmente, estará compuesta por un conjunto de GE.
Desarrollador de *Enablers*	Desarrolladores de componentes o complejos sistemas que pueden ser instanciados para proveer funcionalidad facilitando así el desarrollo de una aplicación. Estos *enablers* son diseñados para ser fácilmente reutilizables con una API claramente definida. La principal diferencia entre una aplicación y un *enabler* es que los usuarios primarios de una aplicación son los usuarios finales, mientras que los usuarios de un *enabler* es una aplicación.
Proveedor de servicios	Son los encargados de desplegar, suministrar y operar las aplicaciones o *enablers*. Este proveedor de servicios suele estar activo en un único dominio de negocio, aunque puede estarlo en varios. (Por ejemplo, estar solo activo en el dominio de la biomedicina y activarse para el dominio de las *Smart cities*)
Proveedor de servicios de alojamiento	Proporciona y opera la infraestructura de alojamiento sobre la cual se despliegan las aplicaciones y *enablers*. Para reducir costes suelen combinarse con el rol de proveedor de servicios. Es importante tener en cuenta que si el servicio de alojamiento ofrecido es en la nube, estamos ante un caso de proveedor de servicios de *enabler*, considerando este *enabler* el almacenamiento en la nube.
Agregador de servicios	Selecciona servicios del conjunto de servicios ofrecidos por los proveedores de servicios para combinarlos,

	creando así un nuevo servicio que ofrezca solución a una necesidad del usuario final.
Proveedor de instancias	Gestiona y crea una instancia a un ecosistema o dominio de negocio específico. Dado un conjunto de *enablers* genéricos define un escenario particular. Proporciona también información sobre cómo utilizar esta nueva instancia y sus términos y condiciones de uso.
Otros roles relacionados con Fiware Lab	Futuros roles relacionados con Fiware Lab (relacionados con los nodos, etc)

Los componentes de la plataforma han sido agrupados en grupos o categorías para facilitar su comprensión y poder seguir un orden de análisis.

En primer lugar encontramos las *Tools* (herramientas) o *developer community and tools architecture*; su objetivo principal es ofrecer un entorno de desarrollo multifuncional que permita el desarrollo y gestión de la construcción de las aplicaciones de internet del futuro. Para ello, se propone integrar toda la experiencia y conocimiento de los desarrolladores sobre kits de desarrollo de software (SDKs), entornos de desarrollo integrado (IDEs) y entornos de desarrollo colaborativos (CDEs) presentándolo de manera integrada y no aislada. Su intención es, por tanto, proveer un único punto de acceso que ofrezca una potente y completa *suite* de desarrollo, la posibilidad de consultar y pedir soporte a la comunidad y la posibilidad de testear, desplegar y monitorizar los resultado finales en cualquier momento y lugar. Podemos concluir que esta *suite* será un conjunto de herramientas, ejemplos de código, documentación, compiladores y librerías que los desarrolladores podrán utilizar. Algunos de sus elementos más destacados son:

- CDE Manager: Este servicio permite crear y gestionar CDEs (entornos colaborativos de desarrollo, estos entornos integrados proporcionan las más típicas características para gestionar y desarrollar un proyecto; incluye herramientas y servicios como un sistema de *tickets*, sistema de control de versiones, *wiki*, SDKs, servidor web…) en una infraestructura virtual. Es posible crear estos CDEs a partir de plantillas predefinidas o empezar desde cero a partir de una máquina virtual vacía. Cuando alguien quiera comenzar el desarrollo de una aplicación este debería ser su punto de entrada.

- IDE: Este IDE se ha desarrollado con la idea de que el usuario puede mejorar su productividad a través del contacto directo con otras herramientas que formen parte de esta suite de herramientas; con esto en mente, se ha reducido al máximo la distancia entre el *manager* del proyecto y los desarrolladores de este.

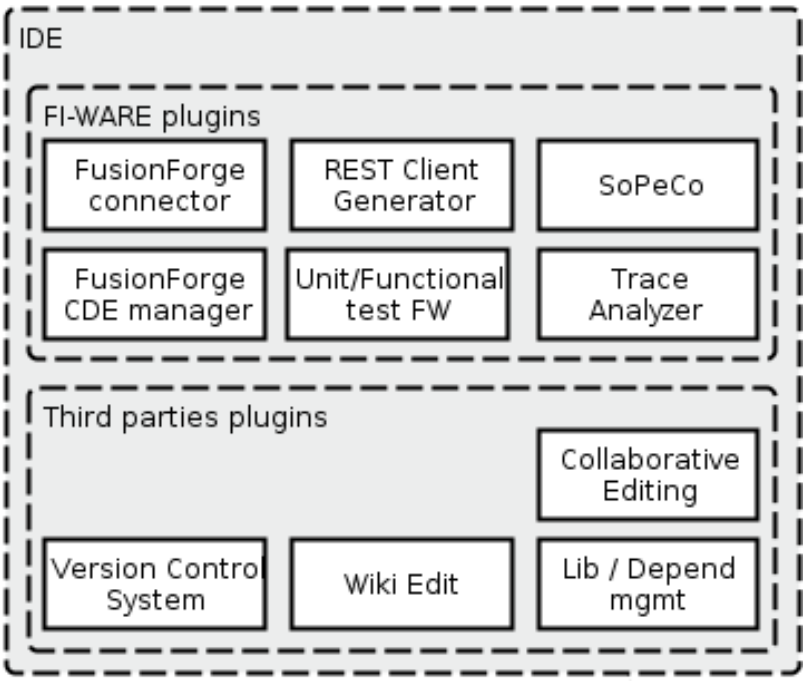


Ilustración 5 Plugins del entorno de desarrollo integrado

El objetivo es producir un IDE basado en Eclipse que contenga todos los *plugins* mencionados.

- eLearning System: Área de aprendizaje para los usuarios finales, interactúa a través de dos canales: Mediante lecciones grabadas sobre diversos temas de interés para el usuario final y mediante sesiones en directo.
- GEI Catalogue: Actúa como contenedor para que los proveedores publiquen sus GEi (Más adelante se tratará el catálogo de GEs disponibles).
- Herramientas de *testing* y validación: En esta categoría se incluye UFT Framework, que simplifica la creación de tests para la validación de código; Trace Analyzer, ayudando en la ejecución de tests de rendimiento durante el desarrollo, despliegue y ejecución de un programa; SoPeCo se encarga de la definición y ejecución de escenarios de tests complejos y analiza los resultados mediante herramientas estadísticas y finalmente PROSA, que monitoriza continuamente el servicio en tiempo de ejecución.

En segundo lugar el *framework* de aplicaciones y servicios. Está compuesto por un conjunto de GE que construyen un ecosistema de aplicaciones y servicios que fomentan la innovación mediante la administración y gestión de aplicaciones y servicios a través de todo el ciclo de vida de estos. Sus componentes (GE) más llamativos son Application mashup que permite a usuarios finales inexpertos crear rápidamente aplicaciones web uniendo widgets y fuentes de datos del catálogo y Data Visualization, que crea visualizaciones de datos, de una manera ágil y sencilla, capaces de trabajar con una gran cantidad de datos heterogéneos.

En tercer lugar Data, o el grupo de gestión de datos. Proporciona un conjunto de GEs extremadamente eficientes que permiten el desarrollo de aplicaciones que requieran la recolección, publicación, proceso y explotación de datos e información en tiempo real a escala masiva. Permite:

- Generar información homogénea a partir de múltiples fuentes de datos.
- Modelar cambios en el contexto como eventos que pueden ser procesados, permitiendo generar nueva información ante la activación de estos.
- Procesar grandes cantidades de información procesándola mediante técnicas de BigData (*Map Reduce* por ejemplo) para generar nuevo conocimiento.
- Gestionar y publicar *opden data*.
- Utilizar los datos existentes para mejorar las aplicaciones.

Todo ello lo consigue gracias a sus GE; recolecta información mediante el Context Broker, procesa metadatos de varias aplicaciones y GEs con Metadata pre-processor, muestra de manera homogénea la información almacenada gracias a Query Broker, realiza anotaciones en la información existente (Semantic Annotation), genera nuevo conocimiento mediante técnicas de análisis empleando el componente Big Data Analysis, reacciona ante cambios de escenario (eventos) ejecutando CEP (Complex Event Processing), permite gestionar contenido multimedia, especialmente video mediante Comp. Video Analysis.

En cuarto lugar Interface to Networks and Device (I2ND), define un espacio que proporciona GEs para ejecutar una infraestructura de red abierta y estandarizada. Con ese fin, provee al usuario de tres funcionalidades:

1. Proporiona un *middleware*[10] de integración avanzada para ser utilizado por los GEs que requieran un gran rendimiento y comunicación.
2. Proporciona elementos para crear productos de gran valor con configuración adaptable (público, privado, negocio).
3. Dota de funcionalidades extra para permitir crear aplicaciones automáticas de una manera más veloz.

La arquitectura de este espacio está formada por cuatro GEs:

1. Connected Devie Interface (CDI): Gestión de los dispositivos conectados (móviles, tablets, ordenadores…) permitiendo por ejemplo control remoto o la comprobación del estado de los dispositivos.
2. Cloud Edge (CE): Se encarga de gestionar los proxies (puertas de enlace que conectan y controlan la configuración de los nodos), estos pueden o no sr accesibles desde el exterior de la red.
3. Network Information and Control (NETIC): Administra las redes abiertas.
4. Service Capability, Connectivity and Control (S3C): Administra las redes ocultas.

En quinto lugar, Internet of Things Services Enablement (IOT) contiene un conjunto de elementos software que permiten la detección o activación de un dispositivo (este dispositivo suele contener el recurso). Engloba dos dominios diferentes: Puertas de enlace y *backends*. Los GEs pertenecientes al primer dominio proveen de funcionalidades de conversión de protocolos e interconexión de redes entre dispositivos, los pertenecientes al segundo ofrecen funcionalidades de gestión de dispositivos y soporte específico a aplicaciones de la Internet de las Cosas.

En sexto lugar llegamos a la categoría Cloud Hosting, esta ofrece los GEs necesarios para diseñar una infraestructura de alojamiento en la nube que sirva para desarrollar, desplegar y gestionar aplicaciones y servicios de la Internet de las Cosas. La arquitectura *Cloud* de Fiware está altamente inspirada en OpenStack, ofrece los siguientes servicios:

- Fiware IaaS GE: proporciona la capa básica de la infraestructura, su objetivo es gestionar los recursos de potencia, almacenamiento y red de la infraestructura. Está compuesto de varios componentes:

[10] Un *middleware* es un software que asiste a una aplicación para interactuar y comunicarse con otras. Añade una capa de abstracción simplificando el trabajo.

        - Fiware Cloud Compute Service: basado en el componente de OpenStack Nova, proporciona y gestiona contenedores de máquinas virtuales así como sus recursos asociados.
        - Fiware Cloud Image Service: basado en OpenStack Glance, permite gestionar imágenes pre-configuradas para instalarlas en las máquinas virtuales (o contenedores Linux).
        - Fiware Cloud Volume Service: basado en OpenStack Cinder, porporciona y gestiona bloques de almacenamiento persistente.
        - Fiware Cloud Network Service: basado en OpenStack Neutron, gestiona las redes virtuales es decir, conecta las máquinas virtuales entre ellas y con las redes exteriores.
        - Fiware Cloud Orchestration Service: basado en OpenStack Heat, permite coordinar el aprovisionamiento y gestión en tiempo real de los recursos (máquinas virtuales, redes, etc) incluyendo itnerdependencias entre ellos
    - Fiware Cloud Object Storage Service: basado en OpenStack Swift proporciona un lugar de almacenamiento escalable, flexible y eficiente para almacenar y recuperar objetos
    - Fiware Cloud Cloud Application Management Service: basado en OpenStack Murano habilita la obtención y gestión de aplicaciones complejas en las máquinas virtuales.
    - Fiware Cloud Cloud Policy Service: Permite definir reglas y ejecutar acciones en respuesta a eventos asociados a dichas reglas.
    - Fiware Cloud Monitoring Service: Permite recolectar y distribuir métricas de recursos asociadas a las máquinas virtuales o hosts.

En séptimo lugar encontramos la categoría Security. Por sus características, los futuros servicios de internet estarán expuestos a diferentes peligros a través de la red. Es un gran reto obtener servicios seguros y de confianza con un impacto mínimo en la eficiencia. El objetivo de Fiware es demostrar que estos servicios son seguros por su diseño. La arquitectura de esta sección está dividida en cuatro bloques de GEs:

1. Cybersecurity: Creado para detectar los riesgos de ciberseguridad. No solo permite detectar ataques, sino que automáticamente genera contramedidas contra estos dependiendo de su contexto. Está compuesto de seis componentes:

- Cyber Data Extraction: Extrae y procesa todos los datos del sistema que puedan ser relevantes para la seguridad (topología de la red, reglas de firewall, resultados de escáneres de vulnerabilidad, etc). La información generada es utilizada por las herramientas Attack Paths y Remediation Engine (descritos posteriormente).
- Attack Graph Engine: Se trata de un analizador de vulnerabilidades que permite generar un grafo con todos los posibles caminos de ataques.
- Scored Path Attacks: Analiza los principales caminos de ataque a partir de grafo de caminos de ataque y computa una puntuación a cada camino en base a la probabilidad de que ocurra y al impacto que generaría.
- Remediation Engine: Ayuda a mitigar los riesgos tomando medidas de seguridad computando las diferentes posibilidades de ejecutar un grafo de ataque y estima un coste para cada uno.
- Visualization interface: Es la interfaz de usuario para gestionar los componentes y analizar dinámicamente los riesgos del sistema de información. Permite visualizar camino de ataque con su puntuación y sus posibles soluciones.
- Privacy-Preserving Data Sharing (P2DS): Permite a una organización compartir los datos e información relativos a un ataque con éxito contra esta.

2. Identity and Acces Management: Gestiona la identificación y permisos de acceso. Está compuesto de tres componentes:
   - Identity Management GE: Proporciona gestión de identidad, autenticación y credenciales a nivel de usuario, organización y aplicación.
   - Authorization PDP GE: Gestiona políticas de autorización en formato XACML[11] y toma decisiones en función de estas políticas para autorizar o denegar el acceso a los solicitantes.
   - PEP Proxy: Juega el rol de punto de refuerzo de las políticas de autorización como un proxy inverso HTTP. Intercepta peticiones al servicio, autentica las peticiones de acceso y, finalmente, autoriza la petición.

[11] "Extensible Access control markup" lenguaje. Es un estándar que define un lenguaje declarativo de políticas de control de acceso.

3. Trust and Trustworthiness: Dota a los desarrolladores de aplicaciones un entorno pada ayuda en el desarrollo de aplicaciones confiables. En concreto, ayuda a conocer y cumplir con los objetivos de confiabilidad; incluye una plataforma de desarrollo dirigida a cumplir los objetivos de confiabilidad y una aplicación de evaluación de certificados para editarlos, firmarlos…
4. Privacy: Proporciona soporte de Privacy-Preserving Attribute-Base Credentials (P2ABC) [38] . Esta tecnología permite a las personas demostrar algo sobre ellos mismos sin revelar ningún otro dato y se ejecuta a través de múltiples protocolos multi ronda que exige el intercambio de numerosos artefactos.

Finalmente, en octavo lugar un conjunto de GRs genéricos contenidos en la categoría Advanced Web-based user Interface. Está formado por un conjunto de GEs que proporcionan una experiencia de usuario avanzada mediante el uso de HTML5 e interfaces de usuario basadas en web. Incluye software para la creación de interfaces web en dos y tres dimensiones así como compatibilidad con realidad aumentada y renderizado en la nube. Se divide en:

- Client Core: Proporciona la funcionalidad básica para crear interfaces basadas en HTML, incluye los GE de 2D-UI y 3D-UI.
- Server Core: Provee de un servidor de sincronización escalable. Mediante el GE Synchronization, se permite instanciar múltiples interfaces de usuario web avanzadas en diferentes clientes sincronizados en tiempo real.
- Supporting Services: Este módulo añade servicios típicos a la hora de crear interfaces web como la obtención de datos mediante geolocalización (Gis Data Provider GE), acceso a datos con estructura de lista con punto de interés para posicionarlos en 2D y 3D (POI Data Provider GE) o servicios de sincronización de renderizado en la nube (Cloud Rendering GE).
- Application Oriented Services: Este último módulo proporciona tres GEs: Interface Designer GE es una herramienta de composición y edición de mundos 3D interactivo. Virtual Characters GE gestiona personajes virtuales animados en escenas 3D. Para terminar, Augmented Reality GE proporciona una interfaz de usuario 3D especialmente diseñado para proporcionarla mediante realidad aumentada.

Se ha utilizado Fiware Labs para realizar las pruebas necesarias para el despliegue de esta aplicación utilizando una cuenta con estatus de cuenta de comunidad. Procedemos

a analizar ahora el entorno de pruebas de Fiware Lab, puntualizando las características que ofrece y las opciones utilizadas para este proyecto. Al acceder, se tiene acceso a 6 categorías: Cloud, Store, Mashup, Data, Account y Help&info.

Empezaremos por la pestaña Cloud. En ella se realizan todas las acciones que utilizan los recursos ofrecidos por la infraestructura de Fiware Lab.

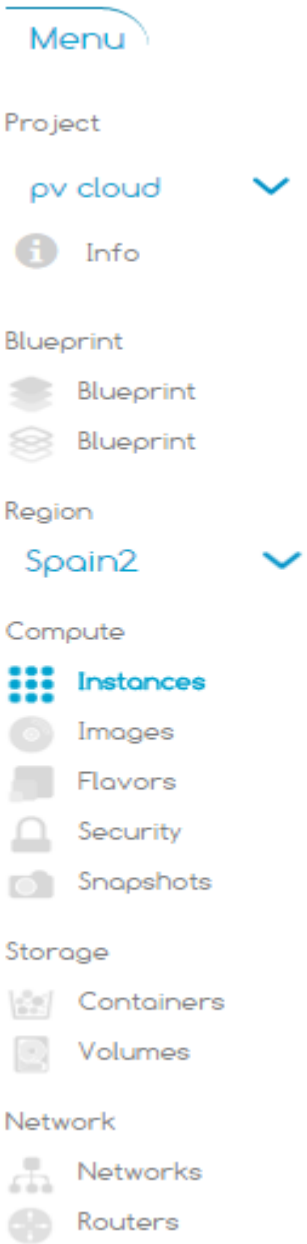


Ilustración 6 Menú sección Cloud de Fiware Lab

El primer elemento del menú (info) corresponde a las credenciales que tenemos asignadas. Estas, son utilizadas para acceder a los servicios de OpenStack desde la

consola de comandos y están formadas por User name, Tenant name[12], Tenant id, Region, Authentication URL.

A continuación el apartado Blueprint[13] nos permite gestionar toda la información relativa a los *blueprints*. En primer lugar, permite crear plantillas de *blueprints*. A estos *blueprints*, o plantillas, se les puede añadir componentes software que serán desplegados (*Tier*). Estos *Tiers* actúan como imágenes independientes y presentan las siguientes opciones de configuración:

- Nombre: nombre que se le va a dar al *Tier*.
- Región: región a la que va a pertenecer el *Tier*.
- Flavor: recursos asignados al *Tier* (CPUs, memoria RAM, etc.).
- Imagen: imagen que va a sr utilizada como base del *Tier* (Ubuntu, Devian, CentOS…)
- Keypair: *keypair* empleada para acceder al recurso desde el exterior.
- Instancias: número máximo y mínimo de instancias del *Tier*, así como su número inicial.
- Software: permite indicar, de un catálogo de software, el software que debe instalarse y desplegarse en el *Tier* cuando se cree una instancia de la plantilla. Esta instalación se realiza de manera transparente al usuario, lo que permite que el usuario pueda crear instancias personalizadas de manera sencilla y rápida.
- Redes: redes a la que pertenece el *Tier*. Puede ser accesible desde el exterior o pertenecer a una red interna.

Una vez creada la plantilla, esta puede ser desplegada para crear así una instancia configurada con los parámetros asignados en la plantilla lista para funcionar. Es importante mencionar que el *blueprint* genera automáticamente un grupo de seguridad (se hablará en detalle de estos más adelante en el documento) con las reglas correspondientes al software instalado.

En la sección *compute* encontramos todos los datos referentes a la infraestructura de Fiware Lab y cómo la utilizamos divididos en cinco subgrupos. El primero de ellos muestra las instancias activas así como sus datos más relevantes y nos permite gestionarlas, conectarnos a ellas y monitorizarlas. El segundo y tercer grupo nos muestran

[12] Grupo de usuarios que tienen acceso a unos recursos. Término homólogo a organización.
[13] Por blueprint se entiende la especificación de una plataforma para ser desplegada.

las imágenes (se tiene la opción de crear instancias a partir de estas imágenes) y *flavors* que podemos utilizar. En security creamos y asignamos IPs, grupos de seguridad (nos permiten definir un conjunto de reglas para determinar los puertos a los que se puede acceder así como limitar el acceso desde un conjunto de ips) y *keypairs*. Por último, esta sección de compute, snapshots, nos proporciona una interfaz para gestionar los snapshots que hayamos realizado tanto a instancias como a volúmenes. Un *snapshot* es una instantánea del estado de un sistema en un momento determinado.

Existen dos secciones más, Storage, con el que gestionar contenedores, espacios de almacenamiento de objetos que permite acceder remotamente a él mediante *cloud data management interface* (CDMI)[14], y volúmenes, espacio de un tamaño máximo de 50 GB que puede ser añadido a las instancias existentes para proporcionar almacenamiento extra, y network, que se encarga de la creación y gestión de redes con las que se configurarán las instancias así como de las puertas de enlace.

A continuación encontramos la Store donde se puede tanto ofrecer como conseguir software. Es la principal encargada de gestionar la venta y ofertas de los GE. Permite la integración de ofertas publicadas por un agregador y puede actualizar un GE presente en la Store para introducir una nueva oferta. Hay que tener en cuenta que actúa como punto final desde el cual es posible descargar estos recursos ya que no almacena el contenido del GE; por ello, la Store define una API que el proveedor de servicios debe implementar y que será usada de manera indirecta para descargar los recursos.

A nivel arquitectónico está dividida en diversos módulos funcionales:

- Módulo de interfaz: módulo responsable de comunicarse con el GE Marketplace para registrar y eliminar instancias del GE Store (El GE Store representa uno de los servicios básicos del ecosistema Fiware; este conjunto de componentes gestionan el ciclo de vida completo de los servicios desde su creación hasta su monetización mientras que el GE Marketplace es una plataforma que, a partir de una o varias Stores, ofrece un catálogo de servicios uniforme a los usuarios para su compra. La transacción se realiza en la Store mientras que el resto del proceso se realiza en el Marketplace).
- Módulo de interfaz de repositorio: Se comunica con el GE Repository (Este GE proporciona una API consistente y uniforme para acceder a las descripciones de

[14] Protocolo con comunicación para administrar y acceder a almacenamiento en la nube

servicio USDL[15]. Un proveedor de servicios utilizará este GE para publicar la descripción de sus servicios) para obtener los documentos USDL asociados a las ofertas publicadas.

- Interfaz IdM: se comunica con el GE Identity Manager para obtener información acerca de los usuarios y organizaciones.
- Módulo de administración: Es utilizado para gestionar el GE Store y registrar instancias de este GE en el GE de Marketplace.
- Módulo de ofertas: Gestiona ofertas específicas de los usuarios así como las compras realizadas a dichas ofertas.
- Módulo de gestión de usuarios: Gestiona tanto los usuarios como sus diferentes roles[16] y privilegios para controlar el acceso a las funcionalidades de la Store.
- Módulo de contrataciones: Gestiona las suscripciones y compras a las diferentes ofertas publicadas en la Store. Es el encargado de contactar con los diferentes sistemas de gestión de pagos, recibiendo la confirmación de los pagos y dar acceso a los usuarios a los servicios contratados.
- Módulo de búsqueda: Ejecuta las búsquedas realizadas por los usuarios para proporcionar las ofertas que cumplan los criterios especificados en la búsqueda.

Permite acceder tanto por la interfaz web de usuario como mediante la API para descargar los GE deseados. Este software se divide en tres categorías: servicios, datos y widgets/Mashups.

Seguidamente llegamos a la pestaña Mashup. Nos da acceso al GE Wirecloud, que está orientado el desarrollo de aplicaciones web *mashup* a usuarios finales que no posean un conocimiento avanzado. Permite elegir de entre un con junto de *widgets* y *mashups* prefabricados del catálogo y conectar los *widgets* fácilmente entre sí para crear un panel de control con funcionalidad RIA[17]. Además, posee un editor dedicado a conectar estos *widgets* con servicios *back-end* o fuentes de datos externas.

Al acceder, nos permite crear espacios de trabajo donde podemos elegir entre una configuración libre o asignándoles una dimensión de filas y columnas. En este espacio disponemos de una serie de *widgets* y *mashups* predefinidos que podremos añadir a nuestro

[15] Del inglés Unified Service Description Language, define un lenguaje para describir tanto la parte técnica como de negocio de un servicio para que este sea localizable y consumible.
[16] Existen cuatro roles: Administrador, proveedor, cliente y desarrollador.
[17] Proviene del inglés rich internet application, una aplicación RIA es una aplicación web que posee muchas características de una aplicación de esritorio.

espacio de trabajo para que sean configurados. Este catálogo de elementos predefinidos puede recibir dos componentes de dos formas distintas: podemos añadir *widgets* creados por nosotros mismos subiendo un archivo wgt (extensión utilizada al generar *widgets*); o acceder al market de Fiware para añadir al catálogo cualquiera de los *widgets* disponibles. Una vez obtenidos, tan solo es necesario arrastrar los *widgets* al espacio de trabajo para que se ajusten automáticamente a la platilla definida anteriormente, cada uno de estos *widgets* son configurados con una serie de parámetros como son la URL a la que acceden al servicio del cual obtienen los datos entre otros. Una vez configurados, estos *widgets* pueden ser interconectados para intercambiar información.

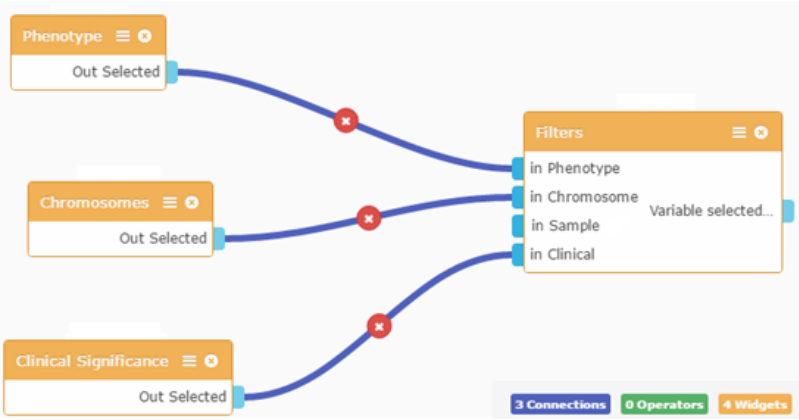


Ilustración 7 Ejemplo de Wiring en Mashup

La siguiente categoría es Data, está implementada sobre CKAN, un sistema gestor de información que hace que estos sean accesibles. Proporciona herramientas para agilizar la publicación, intercambio, búsqueda y utilización de datos. Se subdivide en cuatro apartados:

- Datasets: Permite obtener y filtrar conjuntos de datos publicados por diversos criterios (más información sobre las organizaciones en el próximo punto). Estos conjuntos de datos pueden filtrarse por temática, organización publicadora, formato de los datos (JSON, CSV, XLM…) o licencia con la que fueron publicados. Una vez obtenido el recurso, en un solo clic pueden obtenerse los datos y ser filtrados (si es posible filtrarlos) con la interfaz web que proporciona la herramienta de una forma intuitiva y eficiente.
- Organizaciones: Muestra las distintas organizaciones que han subido contenido, incluye tanto información sobre esta (datos de la organización, actividad, etc.)

como los conjuntos de datos compartidos y si ha realizado alguna petición de datos. Es posible suscribirse a estas organizaciones.

- Grupos: Estos grupos se utilizan para crear y gestionar colecciones de *datasets*. Por ejemplo, un grupo que contenga los datos referentes a un proyecto o que pertenezcan a un tema en particular.
- Peticiones de datos: Los usuarios pueden realizar peticiones de datos que todavía no se hayan publicado en la plataforma. Si bien sobre el papel es una buena opción, no ha habido casi actividad (3 peticiones en un año) y han sido ignoradas.

Esta característica no ha sido utilizada ya que no dispone de ningún dato de carácter genético o médico que pudiera resultar de interés.

Posteriormente la pestaña Account nos muestra, por un lado, las organizaciones de las que formamos parte (o hemos creado) y observar tanto los miembros como las aplicaciones que la organización tiene autorizada utilizar. En nuestro caso, hemos creado una organización con únicamente dos aplicaciones autorizadas: Cloud, que nos permite gestionar todos los recursos *Cloud* (imágenes, instancias, *blueprints*, etc) y Store (Que nos da la posibilidad de publicar los servicios desarrollados y adquirir aquellos que sean de interés). Estas organizaciones pueden entenderse como los editores o desarrolladores de los servicios ofrecidos en la Store. Por otra parte, nos muestra información relativa a las aplicaciones con que estamos relacionados: Aplicaciones desarrolladas por nosotros disponibles en la Store, aplicaciones obtenidas y aplicaciones autorizadas.

Finalmente, encontramos la categoría de Ayuda e información. Esta sección está compuesta principalmente por video-tutoriales subdivididos en las cuatro primeras características citadas (Cloud, Store, Mashup y Account). Cada sección incluye un conjunto de videos muy útiles para familiarizarse con el sistema (instanciar máquinas virtuales, obtener GEs, etc), pero resultan meridianamente insuficientes cuando la duda o necesidad es de una dificultad no trivial; no obstante, Fiware ofrece un foro de ayuda donde poder consultar dudas y un sistema de consultas por correo electrónico para estas cuestiones, donde se suelen resolver las dudas de manera precisa y rápida. Esta sección no contiene ningún otro contenido que merezca la pena ser reseñado.

Se han desarrollado una serie de herramientas que ayudan al despliegue, motinorización y mantenimiento de los nodos de Fiware Labs. Estas herramientas han sido encapsuladas en la suite Fiware Ops tools. Esta suite tiene tres objetivos:

1. Permitir la configuración de la infraestructura desde cero hasta su inclusión en Fiware Lab.
2. Proporcionar servicios para actividades operacionales en materia de infraestructura. Por ejemplo mantenimiento, actualización, etc.
3. Contribuir a la comunidad OpenStack aportando nueva funcionalidad en cuanto al despliegue y operaciones de infraestructuras en la nube.

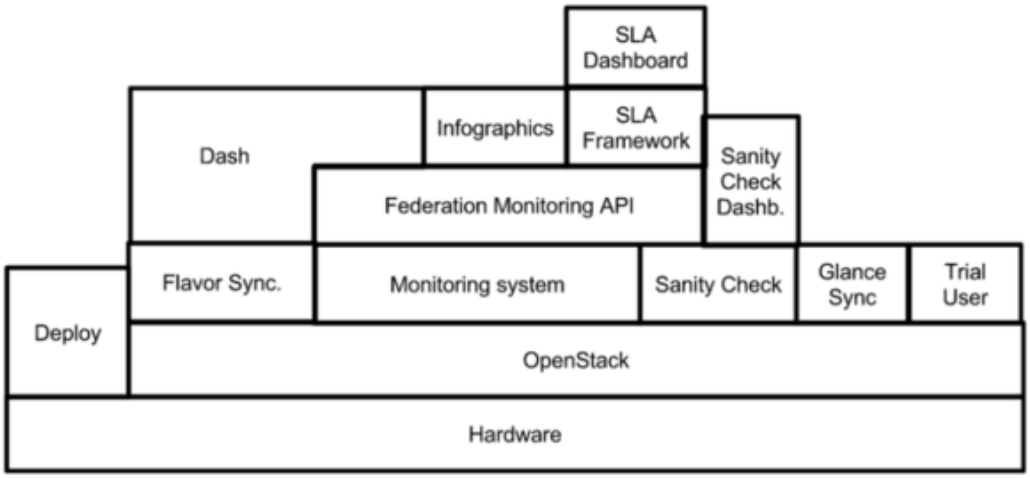


Ilustración 8 Componentes desarollados de la suite Fiware OPS tools

Por ello, dentro de esta suite se ha creado alrededor de cuatro características o funciones [39], [40] :

1. Platform Deployment Tool – OPS-Deploy: Componente de software de código abierto desarrollado por la comunidad de OpenStack. Proporciona una interfaz de usuario web que permite a un administrador desplegar y gestionar intuitivamente instancias en el entorno OpenStack. Se utiliza en la plataforma Fiware para realizar despliegues testeados donde se garantice la correcta configuración de las redes así como una resolución de conflictos más sencilla.

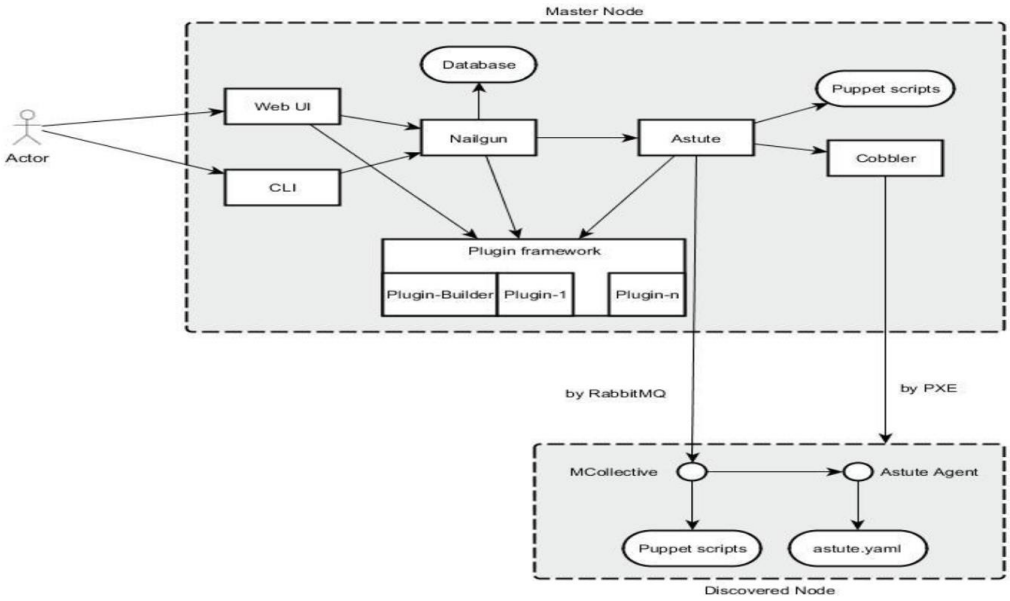


Ilustración 9 Arquitectura de alto nivel

Como podemos observar en la Ilustración 9, está compuesto de tres puppet scripts (estos scripts sirven para gestionar la configuración del componente software), un task orchestrator (componente Nailgun del diagrama que permite ejecutar y modificar el orden de ejecución de las tareas), un task executor (componente Astute encargado de ejecutar las tareas) y una interfaz de usuario. Su objetivo es facilitar el despliegue de nuevos nodos de Fiware Lab basados en OpenStack. Esta arquitectura refleja la estructura de Fuel by mirantis[18], al estar basada en ella.

Se soporta una arquitectura conectable que permite al usuario instalar y configurar prestaciones adicionales para su entorno de una manera flexible. Como se ha podido observar en la Ilustración 9, todos los componentes previamente existentes han sido reconstruidos como plugins. Cada plugin es integrado, mediante el plugin framework, con la interfaz de usuario y activado por Nailgun. Cada plugin interactúa con Nailgun, el cual gestiona los datos de despliegue (datos de la configuración de los discos, de configuración de la red y cualquier tipo de datos específicos del entorno necesarios para finalizar un despliegue).

Astute puede ser interpretado como una composición de los workers de Naiulgun; cada uno de estos workers ejecuta ciertas acciones de acuerdo con las instrucciones existentes en Nailgun. Mientras que Nailgun utiliza bases de datos SQL para almacenar datos e interactúa con sus workers mediante servicio AMQP

[18] Para más información visitar: https://www.mirantis.com/products/mirantis-openstack-software/

(Advanced messaging queue protocol)[19], Cobbler es utilizado como sistema operativo proporcionando servicio DHCP (Dynamyc host configuration protocol)[20]. Finalmente, Puppet es el servicio de desplegué que, mediante Collective, ejecuta tareas específicas como comprobar la conectividad red o el mantenimiento de discos duros.

2. Platform Deployment Tool – OPS-Dash: OPS-Dash se divide en 4 componentes: Fidash (panel de administración y gestión de Fiware Lab), SLA framework y SLA Dashboard (Soporta la configuración del SLA[21] en Fiware Lab) y Maintenance calendar (Permite definir períodos de mantenimiento de las diferentes regiones de Fiware Lab de manera estructurada, facilitando así que esta información llegue al usuario final o a ciertas herramientas).

Ilustración 10 Ejemplo de conexiones entre widgets

Empezando por Fidash, esta es una versión adaptada de WireCloud, lo que la convierte en un *mashup*[22] altamente personalizable permitiendo al usuario definir fácilmente la funcionalidad y el comportamiento del panel; de este modo, este panel está formado por múltiples widgets que el usuario puede utilizar, descardar, añadir o

[19] Se trata de un protocolo de mensajes a nivel de red confiable y eficaz utilizado para crear aplicaciones de mensajes robustas y compatibles con varias plataformas.
[20] Se define así a un servidor que tiene un protocolo de red de tipo cliente/servidor donde el servidor posee una lista de IP dinámicas que las asigna a los clientes según estas van quedando libres.
[21] Acuerdo de nivel de servicio. Se trata de un contrato escrito entre el proveedor de un servicio y su cliente cuya finalidad es fijar la calidad del servicio ofrecido.
[22] Un mashup es un contenido usado de otra fuente mediante una API. Ocurre cuando una aplicación web es usada o llamada desde otra aplicación con el fin de reutilizar su contenido y/o funcionalidad

modificar a conveniencia. Estos *widgets* ofrecen funcionalidad específica (mostrar una lista de máquinas virtuales, por ejemplo) y se basan en APIs ofrecidas por los diferentes servicios de OpenStack u otros servicios creados. Estos *widgets* pueden conectarse entre sí para ofrecer funciones de alto nivel y ofrecer *feedback* basado en las acciones realizadas por otros componentes. Este mecanismo se consigue mediante el uso de un mecanismo llamado wiring consistente en el envío de mensajes asíncronos con información propia (eventos). El usuario de Fidash posee el control de este wiring y puede conectar y desconectar los *widgets* a su voluntad, modificando así el comportamiento del panel. Posee diferentes servicios clasificados en [41] :

- Servicios de OpenStack para gestionar recursos de Fiware Lab. Estos recursos se componen de las máquinas virtuales, permitiendo listarlas, obtener sus detalles, buscarlas, filtrarlas, eliminarlas, iniciarlas y reiniciarlas. Imágenes, permitiendo listarlos, obtener sus detalles, buscarlas, filtrarlas, cambiar su visibilidad y crearlas desde archivo o url remota. Volúmenes, que pueden ser listados, obtener sus detalles, buscarlos, filtrarlos y ser adjuntados a una máquina virtual y flavors, que pueden ser listados, obtener sus detalles, buscarlos y filtrarlos (pueden ser creados y modificados pero se requiere de privilegios de administrador para realizar la acción).
- Monitorización de la infraestructura subyacente, incluyendo CPU, RAM, uso de discos duros y uso de ips.
- Verificación del cumplimiento de los SLA establecidos.

Es importante tener en cuenta que Fidash puede estar compuesto por *widgets* de diversas regiones, es necesario asegurarse de tener acceso a todas las regiones [41]

A continuación pasamos al software relativo al SLA, empezando por SLA Framework. Se trata de una implementación del sistema de gestión del ciclo de vida del SLA compatible con la especificación WS-Agreement (Un protocolo de servicios web para establecer acuerdos entre dos partes; por ejemplo, entre el proveedor de un servicio y el consumidor. Esta especificación está compuesta de tres partes: Un esquema para especificar el acuerdo, un esquema para especificar una plantilla de acuerdo y un conjunto de operaciones para gestionar el ciclo de vida del acuerdo) [42] . Se trata de una aplicación web multiplataforma que permite gestionar el ciclo del SLA desde la creación de la plantilla del acuerdo a la detección de violaciones del

acuerdo. Esta aplicación está basada en un componente desacoplado basado en plugin que puede ser extendido para trabajar en diferentes plataformas. Se utiliza en Fiware combinado con el componente SLA Dashboard para ofrecer una calidad de servicio mínima de los servicios prestados estableciendo un SLA para ellos y registrando la violación de términos del SLA si se producen. Puede operar tanto a nivel de host como de máquina virtual y de servicio. Por el momento, incluye módulos para:

- Definir y anunciar las capacidades de un proveedor de servicios en plantillas de SLA mediante un lenguaje y protocolos.
- Creación de SLA basado en plantillas.
- Monitorizar en tiempo real el cumplimiento del SLA.

Seguidamente aparece el SLA dashboard; este, proporciona una interfaz de usuario web para gestionar los SLA de diversos componentes de Fiware. Es una aplicación Django que permite gestionar todo el ciclo de vida del SLA. Actúa junto con SLA Framework pues mientras que SLA Frameworkd actúa como *back-end*, SLA Dashboard actúa como *front-end*. De este modo, SLA Dashboard permite realizar todas las acciones del SLA Framework de una manera más amigable [43] .

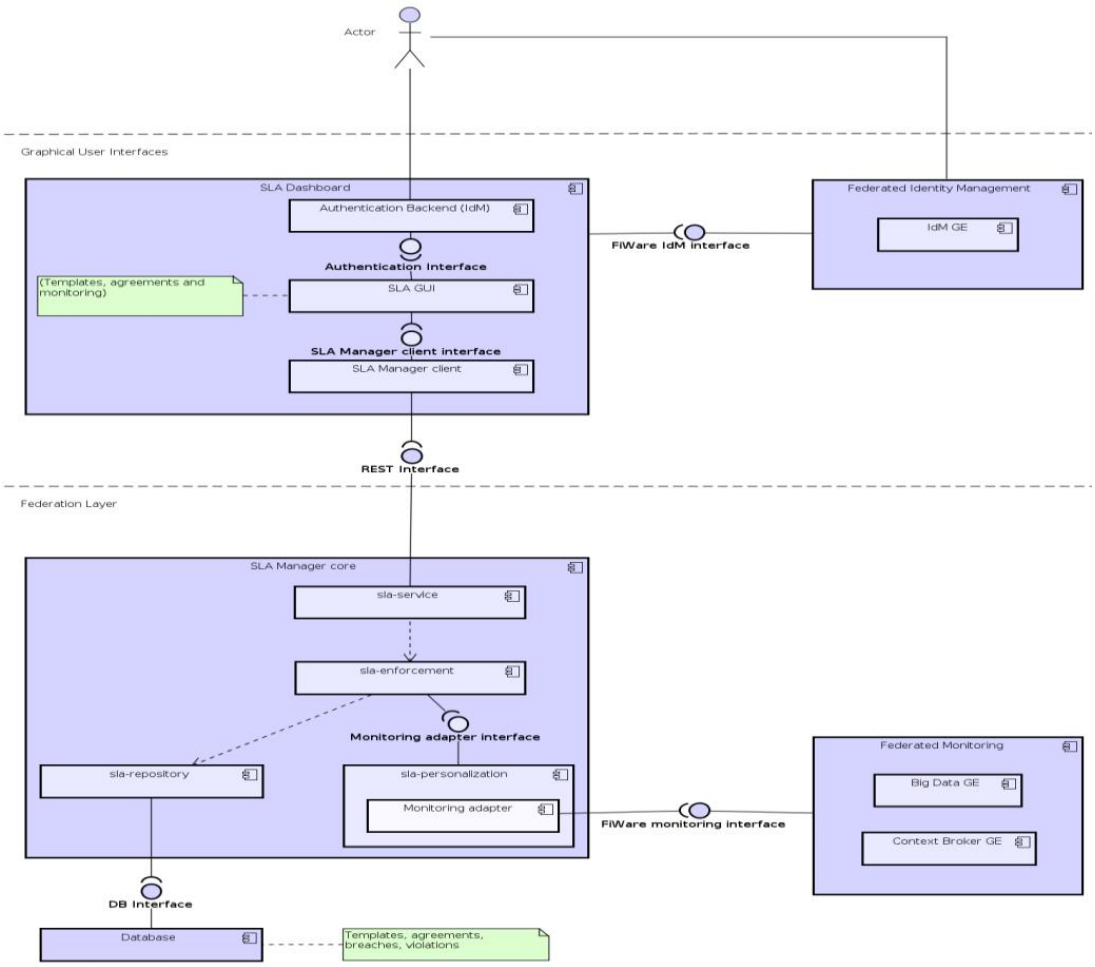


Ilustración 11 SLA Dashboard

La aplicación SLA Dashboard está compuesto de los siguientes directorios [44] :

- Sladashboard: la app relacionada con la aplicación en sí misma.
- Slagui: el proyecto de interfaz gráfica de usuario.
- Slaclient: este proyecto contiene todo el código necesario para conectar con la interfaz REST del SLA Manager así como para la conversión de xml/json a objetos Python.
- Samples: Este directorio contiene los archivos de muestra para realizar tests en SLA Manager.
- Bin: En este directorio se almacenan varios scripts necesarios.

3. dPlatform Analytics Tool – OPS-Health: Son un conjunto de herramientas de código libre para OpenStack que ayudan al usuario a conocer el estatus de cualquier nodo de Fiware Lab. Está compuesto de tres componentes: Sanity Check (incluye tanto el

motor de Sanity Check como su Dashboard), Infogrpahics & Status Page y Federation Monitoring.

El principal objetivo de Sanity Checks es proveer una manera de conocer el estatus de cada nodo de Fiware Lab, está compuesto de una colección de tests que se ejecutan en cada región de Fiware Lab para validar las prestaciones de cada región y su estatus. Para ello, el *framework* de Sanity Checks cubre con sus tests operaciones computacionales, operaciones de red, operaciones de gestión de imágenes y operaciones de almacenamiento de objetos. Estos tests están escritos en Python y se ejecutan sobre el *framework* de *testing*.

Sanity Checks Dashboard es una herramienta gráfica que presenta los datos recolectados por Sanity Checks actuando como *front-end*. Muestra los resultados de todos los *tests* y el estatus de cada región. Permite al usuario reiniciar la ejecución de los *tests* y suscribirse a notificaciones de email para estar al tanto de cualquier cambio de estatus en una región [45] .

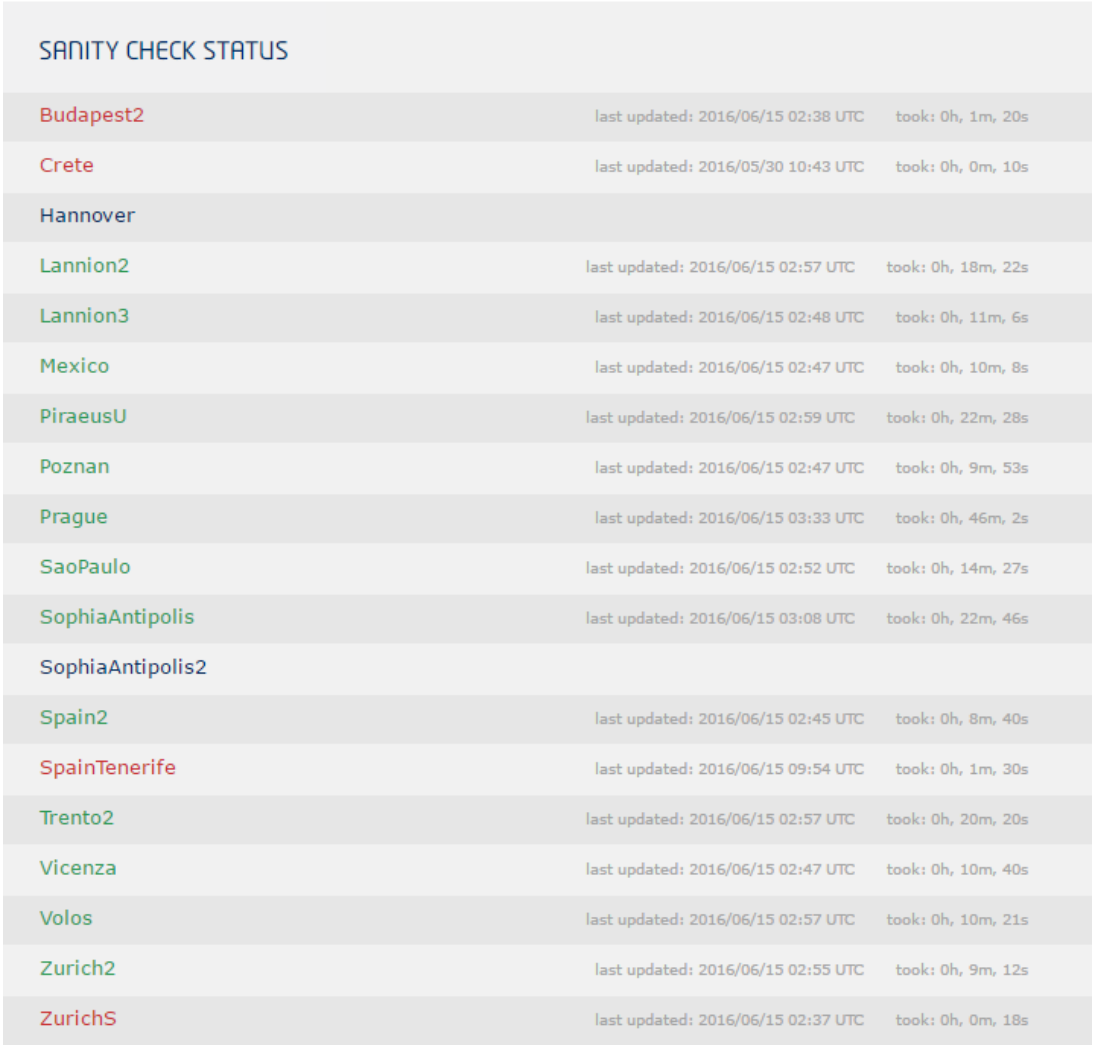


Ilustración 12 Sanity Checks Status

Infographics & Status Page es un servicio simple pero muy importante, permite a los usuarios conocer de una manera intuitiva las capacidades de infraestructura disponibles en la infraestructura de Fiware lab y monitorizar el estatus actual de estos servicios y obtener información de cualquier error existente. Así pues, existen dos servicios: la información referente a la capacidad de la infraestructura de Fiware, más relacionada con el marketing, y un segundo extremadamente importante para conocer el estatus de los servicios ya que permite al usuario conocer las operaciones soportadas por cada región [46] .

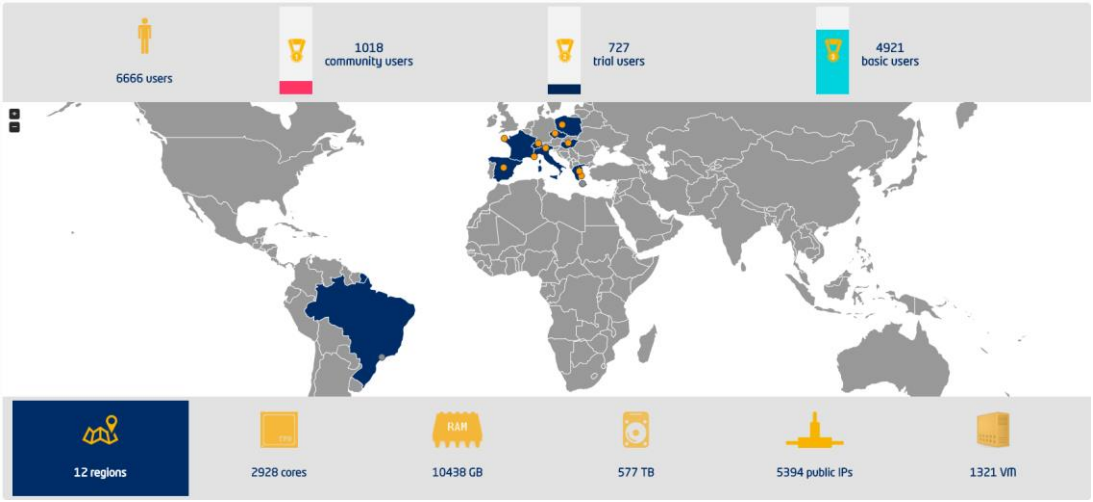


Ilustración 13 Capacidad de la infraestructura Fiware

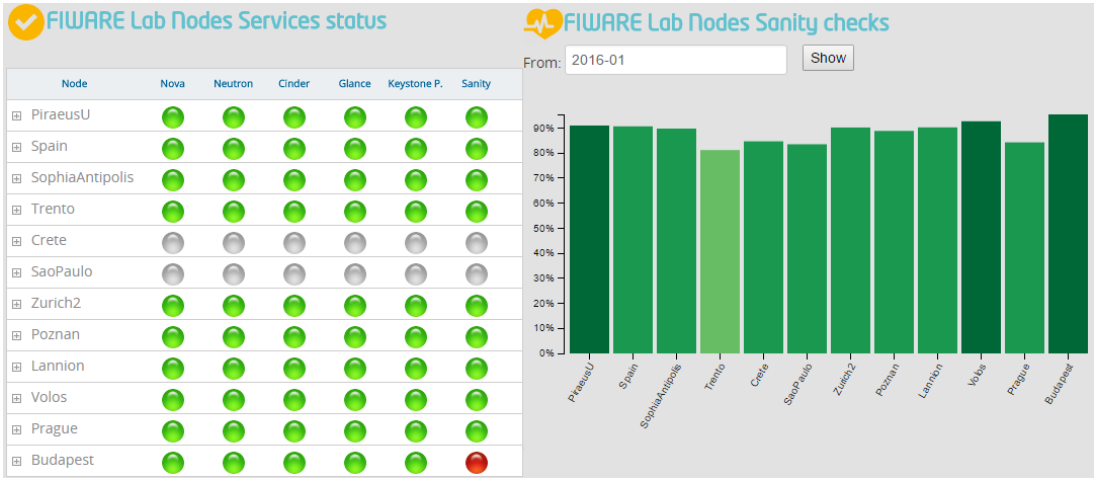


Ilustración 14 Estatus de los Nodos Fiware

Finalmente, Federation Monitoring es una aplicación en NodeJS que proporciona una API que provee de diferente tipo de información relacionada con la instalación de OpenStack, como uso de CPU, ram o disco. Provee información de diversas herramientas y las exporta a los usuarios finales. El acceso a la API está protegido mediante un proxy que evalúa las credenciales del usuario mediante OAUTH2 (protocolo que permite flujos simples de autorización para sitios web).

4. Support Tools – OPS-Toolkit: Provee de un conjunto heterogéneo de herramientas para extender las herramientas presentes en OpenStack, así como añadir nuevos componentes de Middleware para implementar la funcionalidad deseada por las operaciones de un nodo. La mayoría de estas herramientas proporcionan interfaces

de usuario que permiten utilizarse en Fidash. Seis son las principales herramientas que componen este kit:

1. Flavor Sync (Flavor synchronization tool) [47]: Un middleware que actúa como un proxy para gestionar *flavors* en las infraestructuras de OpenStack. Con esta herramienta es posible crear, listar, actualizar y eliminar *flavors* de una infraestructura, además de añadir nuevas características como importar y exportar *flavors*.
2. GlanceSync (Glance synchronization tool) [48]: Herramienta de línea de comandos utilizado para resolver problemas de sincronización de imágenes entre regiones. Sus servidores se sincronizan con distintas regiones tomando como base la región *master*. Existe una región *master* tomada como referencia, a partir de esta, y mediante el uso de metadatos almacenados, se sincroniza con todas las demás regiones. El algoritmo de configuración de Glance es configurable de modo que se pueden cambiar las imágenes que deben ser sincronizadas (por ejemplo, excluyendo las imágenes privadas). Cabe resaltar que GlanceSync puede sincronizar regiones cuyo servidor de Keystone sea diferente del de la región *master*[23]. Las regiones se agrupan por objetivos o *targets*, dos regiones están en el mismo *target* si poseen las mismas credenciales (misma versión del servidor Keystone). El único *target* obligatorio es el *master*, donde está contenida la región *master*. Esta característica implica que la mayor parte de la configuración de Glance se defina a nivel de *targets*. Está configurado para que reemplace imágenes existentes, si se detecta una diferencia de *checksum*[24] se emite una advertencia, el usuario podrá, si lo desea, modificar la imagen para eliminar esta advertencia.
3. Maintenance Calendar tool (backend) [49] : Este middleware actúa como un proxy permitiendo a los encargados de la infraestructura gestionar avisos por mantenimiento. Existen dos tipos de eventos: por una parte, los eventos de mantenimiento asociado a un nodo, que advierten a los usuarios de la realización de tareas de mantenimiento; estos avisos de mantenimiento de nodos serán creados por el propietario de dicho nodo. Por otra parte, se pueden configurar períodos donde no se permita

[23] Diferentes versiones del servidor de Keystone implica un diferente conjunto de credenciales.
[24] Se utiliza esta función para detector cambios accidentales en una secuencia de datos, su función es la integridad de estos.

realizar labores de mantenimiento en la infraestructura; esta acción sólo puede ser realizada por usuarios con permisos especiales (*uptime requester*). Estos eventos serán visibles para todos los usuarios de Fiware. Los usuarios de Fiware pueden listar los eventos existentes, filtrarlos por tipo de evento, fecha de inicio y fecha de fin y obtener información específica de un evento. Los propietarios de los nodos pueden crear y eliminar eventos de mantenimiento mientras que los eventos de período de no mantenimiento pueden ser creados y eliminados por usuarios con el permiso *uptime requester*.

4. User management tool (SkuId) [50]: Este componente es el encargado de gestionar los usuarios con licencia de prueba así como de liberar los recursos reservados para usuarios con licencias caducadas. Cuando esta licencia expira, puede mejorar la cuenta a una de tipo básico. Los recursos que libera son: recursos nova (servidores, *keypairs*, grupos de seguridad), recursos glance (imágenes y *snapshots*), recursos cinder (volúmenes), recursos neutrón (redes, subredes, puertos, ips, grupos de seguridad), *blueprints, templates* y recursos Swift (contendores y objetos).
5. Public keys management tool (Aiakos): Servicio responsable de almacenar las claves públicas correspondientes a cada nodo de Fiware Lab para asegurar el acceso a las máquinas virtuales instanciadas. Cada región debe generar y subir a Aiakos una clave SSH pública y una clave GPG[25] pública.

Tal como se ha visto en la introducción de esta sección, el programa Accelerator de Fiware tiene la finalidad de financiar proyectos de pequeñas y medianas empresas. Los beneficiarios de este programa pueden solicitar una mejora para obtener una cuenta de comunidad (también pueden acceder ciertas compañías y particulares que deseen utilizar Fiware en sus proyectos). Si bien esta mejora es gratuita, el acceso está restringido para poder asegurar una cantidad de recursos óptima y la duración de esta mejora en la cuenta es de 9 meses prorrogables. Para solicitar esta mejora serán necesarios los siguientes datos:

- Usuario: Nombre completo de la persona asociada a la cuenta principal.
- Correo electrónico: El e-mail asociado a la persona responsable del proyecto.

[25] Herramienta de código libre de cifrado y firmas digitales.

- Compañía: La compañía o universidad a la que pertenece la persona responsable. En caso de que sea una universidad es posible incluir el departamento al que pertenece.
- Número de desarrolladores: El número de desarrolladores que se espera vayan a participar en el proyecto.
- Programa Accelerator de Fiware: Indicar el programa Accelerator al que pertenece el proyecto, si pertenece a alguno.
- Nodo de Fiware Lab preferido: El nodo donde la aplicación será desplegada, cada programa Accelerator tiene un conjunto de nodos de referencia [51] .

Programa Accelerator	**Nodos**
CEED Tech	Budapest, Lannion, Berlin
EuropeanPioneers	Karlskrona, Lannion, Berlin
FI-C3	Lannion, Berlin, Prague
frontierCities	PiraeusU, Stockholm, Trento
IMPACT	Spain, Prague, Trento
INCENSe	Waterford, Budapest, PiraeusU
SOUL-FI	Karlskrona, PiraeusU, Zurich, Spain
SpeedUP!	Volos, Karlkrona, Zurich, Spain
FI-ADOPT	Stockholm, Volos, SophiaAntipolis
Finodex	Trento, Spain, SophiaAntipolis
FICHe	Prague, Spain, Volos
FRACTALS	PiraeusN, Poznan, Zurich
FInish	Budapest, Poznan, Gent
SmartAgriFood	PiraeusN, Poznan, Waterford
FABulous	Gent, PiraeusN, SophiaAntipolis
CreatiFI	Spain, Trento, Gent

Tabla 2 Nodos de referencia por programa Accelerator

- Desplegado: Indicar si la aplicación o proyecto ya está desplegado.
- Recursos necesarios: Descripción de los recursos necesarios
- Nombre: Nombre del proyecto
- Descripción: descripción del proyecto incluyendo los GE utilizados.

Finalmente, el programa Mundus aparece con la finalidad de establecer enlaces alrededor de todo el mundo para ayudar a Fiware a prosperar tanto dentro como fuera de Europa. Cuenta con dos objetivos, el primero consiste en promover el uso de Fiware tanto en Europa como en el resto del mundo; el segundo, asegurar que el ecosistema de Fiware es sostenible en el medio y largo término. Siguiendo este propósito, se ha empezado a desarrollar una red global de nodos en todos los continentes: En América, se ha establecido un nodo en México y otro está siendo creado en Chile mientras que se han establecido conversaciones con Argentina, Costa Rica, Panamá, Nicaragua y Colombia; en Asia también ha habido avances cuando en Marzo de 2015 Corea del Sur se comprometió a promover el uso de Fiware como una plataforma común en el desarrollo de aplicaciones de la internet de las cosas; finalmente, en África, tanto Senegal como Mauricio planean crear sus propias regiones de Fiware [52] .

Para el caso de Europa, se han estudiado más de 180 regiones con el objetivo de identificar regiones relevantes que puedan movilizar su ecosistema de innovación para desarrollar nuevas aplicaciones y servicios con Fiware. Para la selección de una nueva región como nodo Fiware se realiza un análisis del compromiso de la región a lo largo de seis pasos tras los cuales, si todo ha sido correcto, la región se convierte en una región Fiware (séptimo paso) [53] :

1. Se identifican las políticas regionales y se comprueba si los negocios presentes en la región con potencial financiador tienen interés en formar parte del ecosistema de Fiware.
2. Se valida si el ecosistema de innovación presente puede adoptar Fiware para el desarrollo de aplicaciones, productos o servicios.
3. Se localiza una estructura u organización capaz de mantener y operar la plataforma Fiware (Convirtiéndose en un nodo de Fiware Lab).
4. Se define un modelo de negocio sostenible del funcionamiento de la plataforma Fiware y su ecosistema (incluyendo costes de implementación, explotación o mantenimiento).
5. Se prepara un documento de oportunidad y se comparte con todas las partes involucradas en el proceso.
6. Se obtiene el apoyo de las partes.
7. Llegados a este punto, la región obtiene visibilidad como una región Fiware, obteniendo así mayor visibilidad mundial, facilitando su acceso a los fondos de

inversión de la Unión Europea además de crear redes con otras ciudades y regiones.

Para crear un nuevo nodo son necesarios una serie de requisitos tanto hardware como de red. De este modo, la mínima configuración requerida para la creación de un nodo es la siguiente: un servidor para almacenar OPS-deploy[26], tres servidores que actúen como cloud controllers[27] (8 núcleos, 16 GB de ram, 1 TB de disco duro), entre seis y nueve servidores que actúen como nodos de computación (200 núcleos físicos, 2GB de ram por núcleo, 20 GB de disco duro por núcleo), tres servidores que actúen como espacio de almacenamiento (8 núcleos, 32 GB de ram y 8TB de disco duro) y un switch[28] de 1 GB como mínimo. Además, debe disponer de conectividad a 1Gbps con una conectividad a usuarios finales de 100 Mbps y disponer de al menos 128 ip públicas disponibles para ser asignadas. Cuando un nodo es creado, es necesario instalar en ellos los siguientes GE: IaaS GE, Monitoring GE, IdM GE, App Management GE [54] .

Una vez creado el nodo, el proceso para unirse a la red Fiware Lab consta de seis pasos:

1. Instalar el nodo. Es posible instalar el nodo Fiware de tres maneras distintas: Se puede hacer una instalación manual (basado en la GEir de IaaS GE), mediante la herramienta FUEL (se trata de una herramienta de código abierto que permite realizar rápidamente despliegues basados en entornos OpenStack) o mediante la herramienta Fiware OPS-Deploy dependiendo el método utilizado en el grado de personalización deseado en el nodo (OPS-deploy y FUEL permiten un despliegue rápido pero con menos opciones de personalización).
2. Configurar el nodo. Se debe confirgurar OpenStack con dos redes externas: La red pública, utilizada para proveer de acceso a internet a las máquinas virtuales, proporciona ips públicas y la red federada, se utiliza como una red externa federada que provee direcciones ip dentro de la VPN de Fiware Lab. A continuación se configuran las cuotas y configuraciones siguiendo en estándar de OpenStack.

[26] Software utilizado en proyectos Fiware para realizar una instalación más coherente y testeada en la federación de nodos de Fiware Lab, garantizando así un despliegue más sencillo y manejable. Para más información visitar: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/OPS-Deploy.
[27] Provee un sistema de administración centralizada para despliegues proporcionando alta disponibilidad. Administra bases de datos, APIs, servicios, etc
[28] Dispositivo que sirve para conectar varios elementos dentro de una red.

3. Testear el nodo localmente. Se debe comprobar el correcto funcionamiento de los grupos de seguridad, instancias, red, etc.
4. Conectar el nodo a Fiware Lab. Una vez se dispone de un nodo autorizado para unirse a Fiware Lab y este ha sido correctamente configurado, se puede coenctar a la red de Fiware Lab tras haber registrado y configurado los servicios del nodo y haber validado el registro.
5. Registrar el nodo en FI-Health. Tan solo es necesario proveer el nombre del nodo y de las redes creadas.
6. Registrar el nodo en Infographics. Se trata de una página que proporciona información en tiempo real de los nodos de Fiware lab.

4.2. Problemas

A pesar de las ventajas supuestamente ofrecidas, y citadas anteriormente, por Fiware, durante el desarrollo de la aplicación nos hemos encontrado con una serie de problemas que merecen ser nombrados y tenidos en cuenta a la hora de realizar un proyecto en el entorno Fiware Lab.

Los primeros problemas han surgido con el uso de los *blueprints*. Lo más llamativo cuando se crea un *template* es que el software que se ofrece está desactualizado a nivel general. Por ejemplo, la versión de JAVA ofrecida es la 7 y la de Tomcat la 6 cuando actualmente las últimas versiones son la 8 para ambas. Cuando se han creado los *Tiers*, cada elemento se supone ofrece una serie de atributos que podemos modificar, aunque en la práctica estos atributos son inexistentes impidiéndonos modificar cualquier posible parámetro; hubiera sido deseable poder modificar al menos la versión del software o las credenciales. Peor aún, como se observa en la ilustración Ilustración 15, resulta el hecho de que al resaltar ciertos elementos en la interfaz web, al observar las instancias de los *blueprints*, estos no muestran información alguna, tan sólo un espacio negro; esto, unido al hecho de que no se muestra la imagen utilizada para crear el *blueprint*, deja una sensación de software inacabado.

Ilustración 15 Información errónea en sección Blueprint

En varias ocasiones, al intentar levantar una instancia de un *blueprint*, se han producido diversos errores, entre los cuales destacan: error 404 indicando que un elemento software añadido no ha sido encontrado, capacidad excedida al intentar levantar la instancia o errores de interacción con OpenStack.

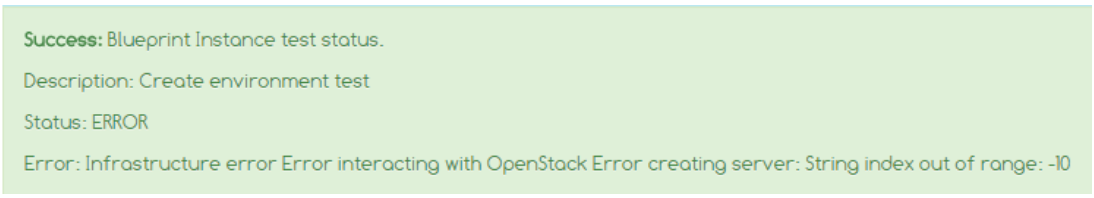


Ilustración 16 Ejemplo de error al crear una instancia

Esto causa frustración al desarrollador pues son errores que, o bien no dependen de él, sino de la plataforma y no puede solucionar, o podrían haber sido evitados con una mejor implementación; además de ofrecer un conjunto limitado de software desactualizado, este no siempre está disponible y no es instalado en el despliegue de la instancia. Asimismo, cuando una instancia falla al intentar levantarse, ni intenta recuperarse ni se elimina para liberar los recursos. Cabe destacar una serie de mejoras que serían más que deseables en la interacción con estos *blueprints*:

- Al crear una plantilla de *blueprint*, no se informa de ninguna limitación a la hora de indicar un nombre para ella (por ejemplo, el uso de guiones bajos), pero cuando se finaliza el proceso de configuración de la plantilla y se crea con un nombre no válido, esta no se crea, cierra la pantalla de creación y nos indica mediante un mensaje por la salida estándar que el formato del nombre no es correcto, perdiendo toda la configuración realizada durante el proceso

y teniendo que empezar desde el principio por unas limitaciones de formato no especificadas en ningún punto.

- Ya sea una plantilla o una instancia, no se permite modificar el nombre una vez creado el recurso.
- Al crear una instancia, se crea automáticamente un grupo de seguridad con las reglas correspondientes, pero con un nombre generado automáticamente, sin darnos opción a modificarlo y, como ocurre con las instancias y plantillas, no se puede modificar el nombre del recurso una vez creado.
- Al tratarse de un catálogo cerrado, no nos permite añadir software propio a una instancia, sumado al hecho de que tampoco podemos añadir imágenes propias, da como resultado una funcionalidad realmente limitada ya que no podemos crear *blueprints* a partir de imágenes propias, ni copias de seguridad.

Al contrario que con las plantillas de *blueprints*, no es posible clonar grupos de seguridad para agilizar el trabajo de creación de nuevos grupos de seguridad ni puede haber más de un grupo de seguridad por instancia.

Al final se ha optado por crear un *blueprint* con el software básico, prescindiendo del que no pudiera ser instalado en el despliegue de la plantilla (postgres, Python, etc.) y modificar el contenido de la instancia directamente accediendo a esta mediante la línea de comandos.

Tras superar los problemas relacionados con los *blueprints*, llegamos a la gestión de las instancias. En cada instancia existen una serie de instrucciones para conectarse:

1. Obtener la *keypair* y cambiar su visibilidad para que no sea pública.
2. Acceder mediante ssh con el comando "ssh –I my_keypair.pem root@public_ip"
3. Acceso como usuario root a la máquina virtual

No obstante al intentarlo se recibe un aviso prohibiéndonos el acceso como root e indicándonos que el acceso se debe realizar utilizando el usuario "ubuntu". Así, la secuencia real es la siguiente:

1. Obtener la *keypair* y cambiar su visibilidad para que no sea pública.
2. Acceder mediante ssh con el comando "ssh –I my_keypair.pem root@public_ip"
3. Obtener mensaje de error.

4. Acceder mediante ssh con el comando "ssh –I my_keypair.pem ubuntu@public_ip"
5. Acceso como usuario "ubuntu" a la máquina virtual.

Esto plantea la problemática de que no hemos accedido como usuario "root" (ya que no se permite) y no disponemos de la contraseña para convertirse en root. Esta contraseña no se proporcionad en ningún lugar de la interfaz web y la solicita tanto para convertirse en usuario "root" como para acceder a la máquina virtual mediante VM display (esta utilizdad permite conectarse a la máquina virtual desde la web de Fiware. Esta aplicación ha dado) desde la interfaz web; esta utilidad (VM display) ha fallado en conectarse a la máquina virtual en el 50% de las ocasiones que se ha intentado.

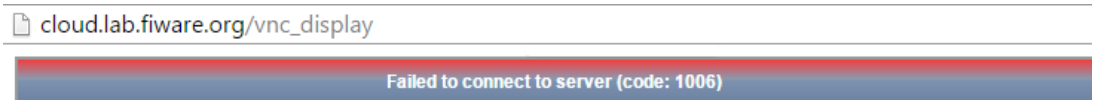


Ilustración 17 Error conectando a VM display

Para poder instalar el software necesario de manera manual sin hacer uso de la contraseña de "root" se ha tenido que seguir la siguiente secuencia:

1. Acceder como usuario "ubuntu" mediante el comando "ssh –I my_keypair.pem ubuntu@public_ip".
2. Cambiar a usuario "root" mediante el commando "sudo su -".
3. Instalar el software necesario.

Por otra parte, ha sido imposible crear y gestionar contenedores mediante la línea de comandos utilizando las credenciales obtenidas por Fiware. Esto es debido a que, al contrario de lo que se indica en la documentación, la región "Spain2" no tiene instalado cdmi[29] y sus *endpoints* están desactualizados.

El siguiente conjunto de problemas está relacionado con la configuración de los recursos en sí. Fiware es un sistema cerrado, la integración que ofrece no es una integración completa ya que se limita a los datos, los GE son componentes cerrados y separados. Cuando se generó la segunda pantalla (el detalle de la composición de la interfaz se explica en los próximos capítulos), esta estaba compuesta con Wirecloud no

[29] Siglas de Cloud Data Management Interface; define una interfaz funcional para realizar operaciones CRUD a conjuntos de datos alojados en la nube.

obstante, no se trata de un *mashup* real ya que Wirecloud permanece en Fiware y es consumido mediante un iframe[30]. Al estar dentro de la plataforma Fiware, la conexión entre los widgets de Wirecloud y el servicio de recuperación de datos requería una conexión HTTPS por lo que se procedió a generar un certificado y cifrar la conexión; esto resultó inútil ya que el certificado estaba autofirmado, no firmado por una entidad certificadora.

Ante esta disyuntiva, se optó por intentar descargar el GE de Wirecloud para instalarlo directamente en la instancia y evitar el uso del iframe. El GE ofrecido por el market está desactualizado y presentaba una serie de incompatibilidades con el software instalado en la instancia por lo que la solución adoptada consistió en descargar Wirecloud de su página oficial, evitando Fiware.

La instalación de Wirecloud se guía mediante un archivo equivalente al pom de Maven, que busca e instala las versiones de los requerimientos iniciales. Estas versiones fueron modificadas para hacerlas compatibles con las existentes en la instancia de Fiware (Python y compresor). A pesar de incluir Wirecloud en la instancia se siguió requiriendo una conexión HTTPS; esto se solucionó utilizando el navegador Firefox y un *plugin* que permite aceptar conexiones no seguras a pesar de que se requiera una conexión segura. Queda pendiente como futuro trabajo el obtener un certificado válido y establecer las conexiones HTTPS entre Wirecloud y el servicio de recuperación de datos.

Además del problema ya nombrado, las instancias de Fiware incluyen un script que automáticamente reinicia la configuración del software instalado. Es el caso por ejemplo de Apache: Cada vez que se modificaban los archivos de configuración para añadir usuarios y darles permisos de gestión, estos archivos volvían automáticamente a su estado original.

Finalmente, sería reseñable el hecho de que por debajo de cierta resolución el menú lateral de Fiwaer lab desaparece y no es accesible, dejando al usuario "atrapado" en la pestaña actual dando lugar a una interfaz no adaptable.

[30] Un iframe es un elemento HTML que permite insertar un documento HTML dentro de otro.

5. Arquitectura

Como ya hemos visto, Fiware nos ofrece una serie de ventajas pero que condicionan la arquitectura empleada para la aplicación. Uno de los objetivos de este proyecto reside en utilizar única y exclusivamente la plataforma Fiware y los componentes que ofrece para desplegar la aplicación.

La arquitectura de la aplicación se divide en los siguientes componentes:

- Base de datos: Base de datos relacional en lenguaje PostgreSQL que contiene los datos. Para obtener las variaciones es necesario implementar un mecanismo de interoperabilidad que permita importar los datos de las distintas fuentes disponibles adaptándolas al modelo propio. Este mecanismo es especialmente importante ya que las fuentes de datos pueden ser muy variadas: Bases de datos relacionales, documentos de texto, documentación HTML, etc.
- Servicios: Componente que resuelve la lógica de negocio para procesar los datos genómicos obtenidos a partir de la base de datos.
- Interfaz: Implementada utilizando el *generic enabler* de Wirecloud y librerías de Javascript, se ha creado una interfaz que visualizará los datos obtenidos desde los servicios proporcionando mecanismos colaborativos y distintos métodos de interacción (ratón, táctil…). Dichos mecanismos colaborativos se obtienen permitiendo que cada participante pueda interactuar con los datos y con los dispositivos de los demás participantes mediante el uso de su dispositivo. Para este componente se ha utilizado tecnología HTML para así asegurar una buena interoperabilidad que permita utilizar esta aplicación tanto en ordenadores como teléfonos inteligentes, tabletas, televisores y cualquier dispositivo con compatibilidad para utilizar un navegador.

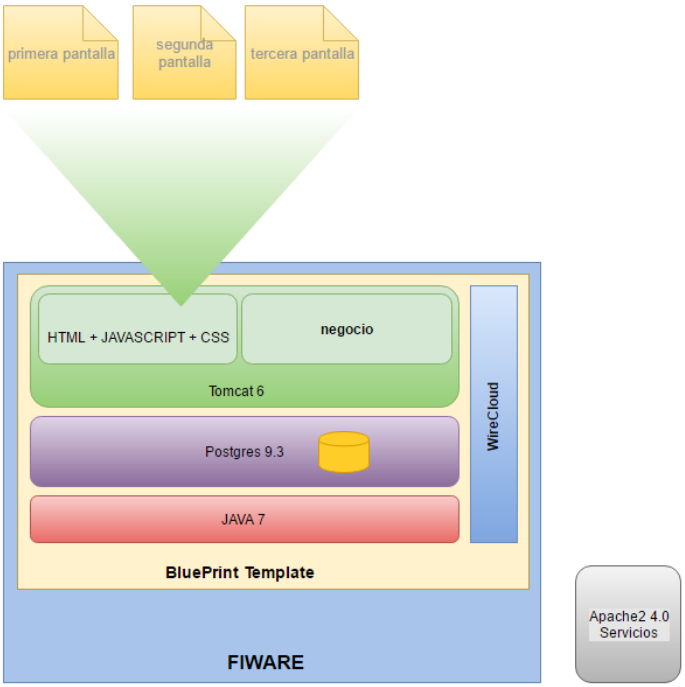


Ilustración 18 Arquitectura

Fiware es la plataforma *cloud* que contiene la aplicación desarrollada y actúa de punto de acceso único. La aplicación está contenida en un *blueprint* cuya plantilla ha sido configurada para que incluya el software JAVA, PostgreSQL y Apache Tomcat preinstalado. Dicho *blueprint* está dividido en 4 componentes:

1. El software JAVA instalado.
2. La base de datos empleada para almacenar la información utilizada en los servicios de recuperación de datos de la aplicación
3. El componente de Fiware Wirecloud para la creación y configuración de widgets utilizados en la segunda pantalla
4. Toda la lógica de presentación y elementos gráficos, desplegados en Tomcat, a los que accede el usuario desde su navegador

Cada instancia de Fiware se comunica con el servidor que contiene los servicios para la recuperación de datos; se trata de un servidor apache donde se ha desplegado una aplicación Spring empaquetada con Maven en formato war (En el capítulo de implementación se trata en detalle el motivo por el cual se ha decidido mantener los servicios como un sistema independiente de la plataforma Fiware).

Dicha aplicación será usada para trabajar en proyectos que integren los datos (tanto públicos como privados) como su visualización y los expertos involucrados en el proceso de diagnóstico. Toda la funcionalidad estará disponible para los usuarios mediante un *dashboard* responsivo donde se presenten las visualizaciones descritas anteriormente; siendo estas configurables en tiempo de ejecución. Los mecanismos de importación de datos incluyen la utilización de datos privados integrados con el material ofrecido por las fuentes externas una vez han sido integradas al esquema interno de la base de datos. Los usuarios pueden crear sus propias visualizaciones empleando los datos disponibles utilizando gráficos preconfigurados o, si lo prefieren, creando sus propios gráficos de un conjunto predefinido. Estos gráficos proporcionan información estadística o relevante de un conjunto de variaciones y, dependiendo del dispositivo donde se visualicen estos, se adaptan a él. A partir de toda esta información, mediante el proceso de análisis y filtrado de datos los usuarios de la aplicación pueden establecer que información es relevante.

La mayor parte de los datos se obtienen de fuentes de datos abiertas que millones de variaciones documentadas para poder comparar respecto a los análisis propios. Debido a la continua evolución del campo, existe un gran número de bases de datos a tener en cuenta:

- Emsemble Variation: Almacena variaciones y, cuando está disponible, enfermedades asociadas a dicha variación e información fenotípica. Dichas variaciones son la principal fuente de información para el soporte al diagnóstico.
  www.ensembl.org/
- Reactome: Esta fuente provee de herramientas intuitivas para la visualización, interpretación y análisis para el análisis del efecto que tienen las variaciones.
  www.reactome.org/
- Gene Ontology: Proporciona una terminología ampliamente utilizada para describir los datos. Se ha considerado esta ontología para el esquema propio de la base de datos.
  geneontology.org/
- dbSNP: Esta base de datos cataloga pequeñas variaciones en secuencias de nucleótidos para un amplio rango de organismos. Es un valioso recurso para descartar variaciones comunes que no son relevantes para el proceso de diagnóstico.
  www.ncbi.nlm.nih.gov/SNP/

- Clinvar: Se trata de un archivo público de relaciones entre variaciones y fenotipos. Proporciona conocimiento acerca de la relación entre una variación humana y los efectos que causa proporcionando un su histórico. Proporciona además investigaciones publicadas para ayudar al proceso de toma de decisiones. www.ncbi.nlm.nih.gov/clinvar
- The Human Phenotype Ontology: Proporciona un vocabulario estandarizado para las anomalías fenotípicas presentes en las enfermedades humanas. De esta base de datos se ha obtenido el glosario de términos con la intención de integrar la información de múltiples fuentes de datos relacionada con una misma enfermedad.
  https://human-phenotype-ontology.github.io/
- UCSC Genome: Provee de secuencias de referencia del genoma humano, así como información sobre genes, proteínas, etc. para comparar las muestras.
  https://genome.ucsc.edu/

Finalmente, pasamos a centrarnos en la arquitectura diseñada para la aplicación. Se ha utilizado un modelo de tres capas cuyos componentes describimos a continuación.

5.1. Modelo de la base de datos

En primer lugar se describe el modelo seguido para diseñar, dando paso a la implementación realizada.

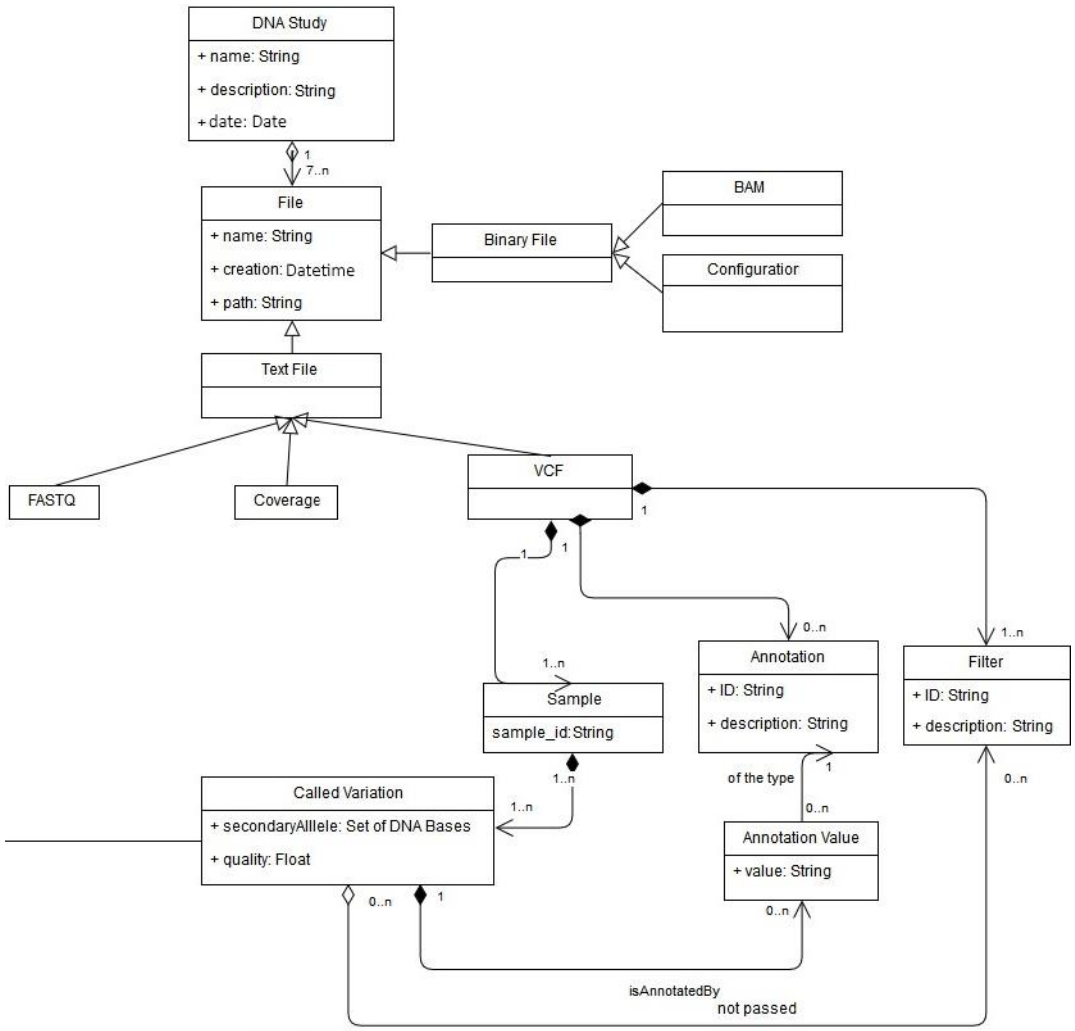


Ilustración 19 Diagrama diseño base de datos 1

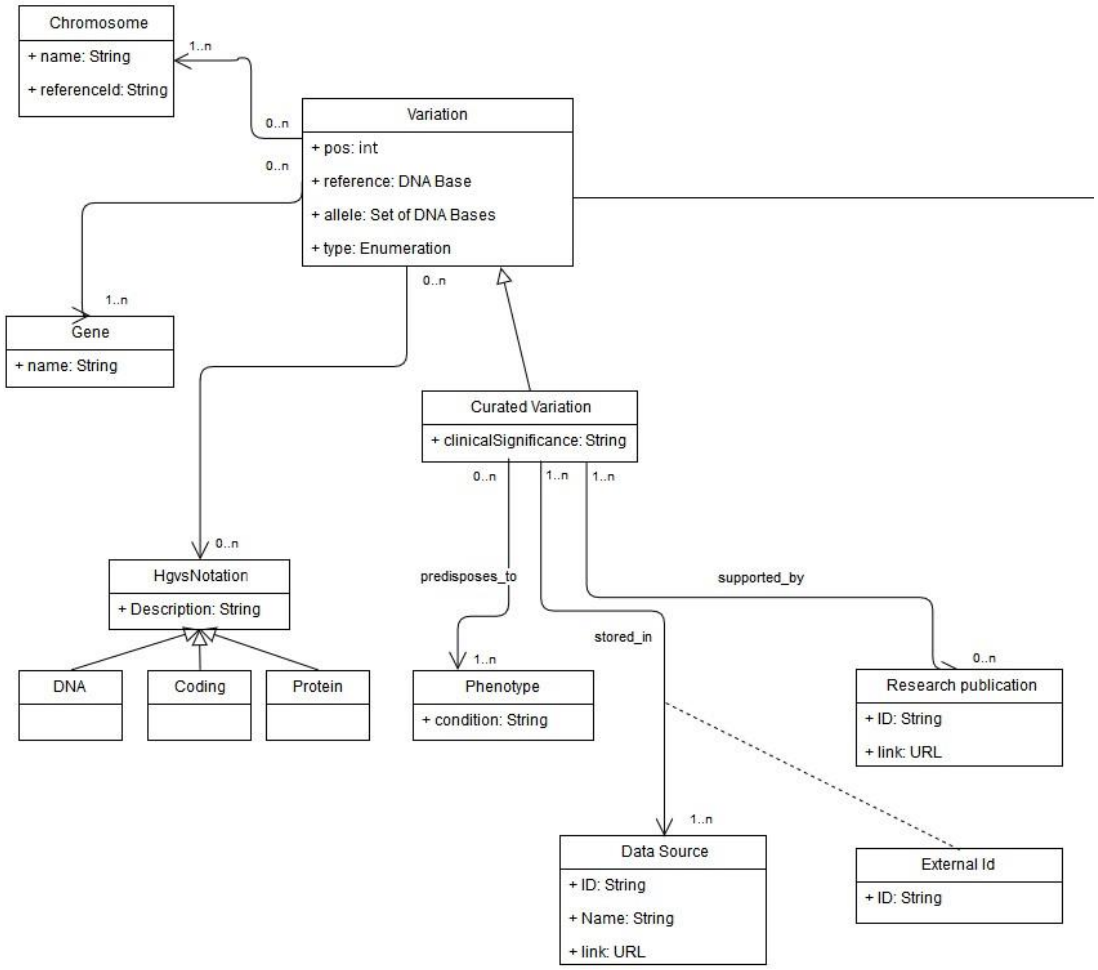


Ilustración 20Diagrama diseño base de datos 2

- DNA Study: Representa un conjunto de archivos como resultado de los distintos procesos de secuenciación. Un DNA Study está formado de al menos 7 archivos (File): Un archivo FASTQ[31], un archivo BAM[32], un archivo VCF[33], un archivo de configuración, un archivo VCF extendido con información propia y dos archivos que representan la cobertura

Nombre	Tipo	Ejemplo	Descripción
Name	String	"Estudio 1"	Nombre del estudio
Description	String	"Análisis principal"	Descripción del estudio que añade información adicional
Date	Date	"10/08/2015 14:00"	Fecha con la que el análisis fue iniciado, corresponde con el primer archivo generado

[31] Se trata de un formato basado en texto para almacenar tanto secuencias biológicas como sus puntuaciones de calidad. Su puntuación de calidad indica la probabilidad de que una base sea incorrecta.
[32] Archivo binario basado en un archivo SAM, Un archive SAM consiste en un archive que contiene alineamientos de secuencia.
[33] Archivo de texto utilizado para almacenar variaciones de secuencias de genes.

- File: Se trata de una entidad abstracta que representa los diferentes tipos de archivos utilizados. Define un conjunto de atributos para el resto de tipos de archivos. Las entidades Binary y Text file están definidas en el modelo por motivos de legibilidad pero no proveen información adicional.

Nombre	Tipo	Ejemplo	Descripión
Creation	Datetime	"10/08/2015 14:00"	La fecha en que el archivo fue creado en el sistema
Path	String	"/home/analysis/"	Directorio en que el archivo está guardado
Name	String	"Sample1.vcf"	Nombre del archivo

- FASTQ: Archivo de texto que, utilizando el formato FASTA, almacenan secuencias de ADN y su calidad asociada siguiendo el estándar Illumina.
- BAM: Archivo binario que almacena alineamientos de una secuencia de referencia. Se trata de una representación binaria del archivo SAM.
- Configuration: Archivo binario generado por la herramienta MYSEQ para almacenar la configuración del *workflow* para ejecutar un análisis.
- Coverage: Se trata de dos archivos, uno de ellos siguiendo el formato gff[34], que representan la confiabilidad de las regiones secuenciadas en base a la cantidad de lecturas realizadas.
- VCF: Archivo de texto que contiene el conjunto de variaciones genéticas estructurales[35].
- Sample: Entidad que almacena la información de cada muestra secuenciada y comparada en el VCF.

Nombre	Tipo	Ejemplo	Descripción
Sample_id	String	"Sample1"	Identificador de la muestra

- Variation: Representa un cambio en el ADN.

Nombre	Tipo	Ejemplo	Descripción
Pos	Int	9161	La posición de la variación, donde la primera base tiene la posición 1 de acuerdo a la secuencia de referencia usada en el alineamiento.
Reference Allel	String	"AAA"	Lista de nucleótidos de la secuencia de referencia para la posición dada.
Alternative Allele	String	"CG,CA"	Lista de posibles alternativas presentes en al menos una muestra.

[34] Para más información visitar: http://www.sequenceontology.org/gff3.shtml
[35] Para más información visitar: http://samtools.github.io/hts-specs/VCFv4.2.pdf

Type	String	Insertion	El tipo de la variación dependiendo del cambio realizado con respecto a la secuencia de referencia

- Called variation: Esta entidad representa una variación detectada en una de las muestras secuenciadas. Cada variación se almacena en una línea de un archivo VCF. Si la variación es heterocigótica se indica el valor del alelo secundario; si por el contrario es homocigótica, el alelo secundario es nulo.

Nombre	Tipo	Ejemplo	Descripción
Secondary Allele	String	"AAA"	Valor del segundo alelo de la muestra cuando esta es heterocigótica
Quality	String	41.01	Puntuación de calidad para la variación.

- Curated Variation: Representa un tipo de variación que ha sido previamente estudiada por genetistas. Contiene información sobre los efectos de la variación.

Nombre	Tipo	Ejemplo	Descripción
Clinical Significance	String	"Lethal"	Evaluación de la importancia de la variación.

- Chromosome: Representa una unidad de ADN que contiene material genético. Se utiliza para localizar y describir variaciones.

Nombre	Tipo	Ejemplo	Descripción
ReferenceId	String	"NC_0000123"	Identificador único usado para referenciar el cromosoma por parte de la comunidad científica
Shrot name	String	"chr1"	Identificador textual para el cromosoma

- Gene: Representa una unidad de ADN que codifica una función específica. Se utiliza para saber que función es afectada por una variación.

Nombre	Tipo	Ejemplo	Descripción
Short name	String	"BRAC1"	Identificador textual para el gen

- Annotation: Esta entidad representa una anotación para incluir información adicional sobre una variación presente en un VCF.

Nomre	Tipo	Ejemplo	Descripción
Shrot name	String	"Annovar-sift"	Identificador textual
Description	String	"Is the SIFT score provided by Annovar to the variation"	Descripción detallada acerca de cómo se ha obtenido el valor de la anotación

- Annotation value: Representa el valor de una anotación para una variación

Nombre	Tipo	Ejemplo	Descripción

Value	String / Int	"PRKCZ" / 1	Para una notación textual representa el valor aportado desde un archivo VCF. Para las enumeradas el id del item

- Filter: Describe los tipos de filtros aplicados a las variaciones de tipo Called Variation. Para describir que filtro no es superado por cada variación, existe una relación entre esta nueva entidad y la variación de tipo Called Variation cuyo nombre es *not passed*.

Nombre	Tipo	Ejemplo	Descripción
Short name	String	"Q10"	Nombre corto y reconocido por los genetistas para reconocer un filtro
Description	String	"Quality below 10"	Explicación del criterio empleado en el filtro

- HGVS Notation: Se trata de una notación estándar para describir variaciones.

Nombre	Tipo	Ejemplo	Descripción
DNA	String	"chr1:g.12345678A >C"	Descripción de la variación a nivel de ADN según la notación HGVS
Coding	String	"NM_00012.2:c. 123A>C"	Descripción de la variación a nivel de código según la notación HGVS
Protein	String	"NP_0000234:p.p. Cys2Ter"	Descripción de la variación a nivel proteico según la notación HGVS

- Phenotype: Esta entidad contiene la información relativa a los efectos que posee una variación

Nombre	Tipo	Ejemplo	Descripción
Condition	String	"Breast Cancer"	Nombre de la condición o enfermedad producida o posiblemente producida por la variación

- Research publication: Provee información sobre las evidencias que demuestran la relevancia de una variación y su relación con ciertas condiciones.

Nombre	Tipo	Ejemplo	Descripción
Id	String	"25741868"	Identificador textual para una publicación de investigación relacionada con una variación
Link	URL	"http://www. ncbi.nlm.nih.gov /pubmed/25741868	Enlace para acceder a dicha publicación

- Data_source: Provee información sobre la base de datos en que se localiza una variación de tipo Curated Variation y su información.

Nombre	Tipo	Ejemplo	Descripción
Id	String	"dbSNP"	Identificador textual de la base de datos

Name	String	"The SNP database"	Nombre completo de la base de datos
Link	URL	"https://www.ncbi.nlm.nih.gov/SNP/"	URL de la base de datos

- External ID: Esta entidad provee información adicional sobre dicha variación de tipo curated Variation en una base de datos

Nombre	Tipo	Ejemplo	Descripción
ID	String	"25741868"	Identificador textual de una variación en una base de datos

Una vez diseñado el modelo conceptual del modelo, se ha creado la correspondiente base de datos utilizando PostgreSQL. Por motivos de usabilidad, se ha diseñado una estructura tan sencilla como fuera posible utilizando un punto intermedio entre el modelo conceptual y la estructura de la información disponible (Archivos VCF).

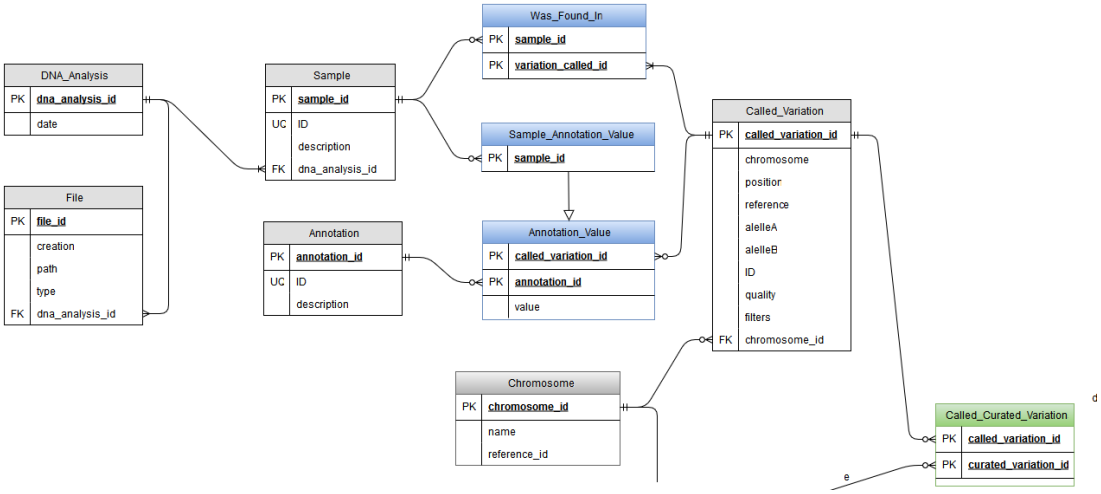


Ilustración 21 Base de datos en PostgreSQL 1

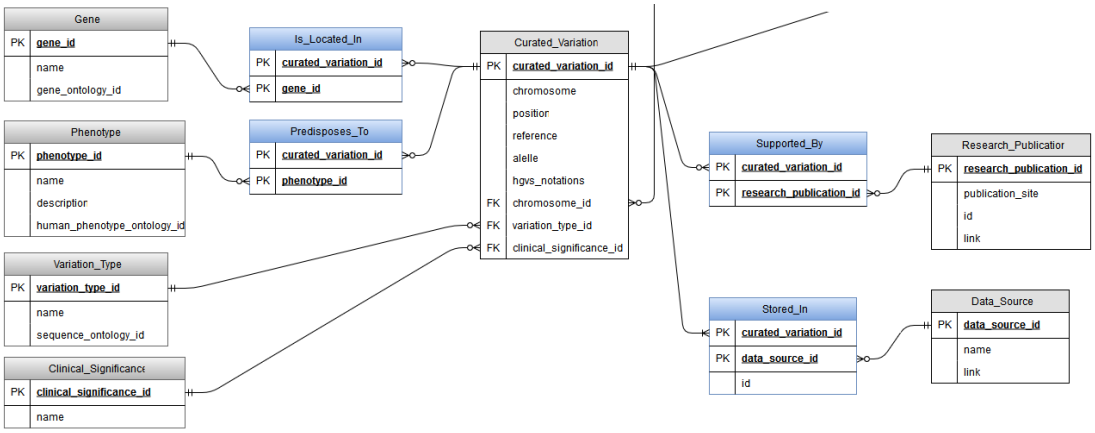


Ilustración 22 Base de datos en PostgreSQL 2

5.1.1. Vista

Se ha decidido crear una vista para la interacción entre la lógica de la aplicación y la base de datos. En dicha vista se han añadido todos los atributos necesarios en la interacción entre el usuario y la aplicación. Cabe destacar que en este primer acercamiento al problema se ha optado por dar unicidad completa a las variaciones; de modo que una variación aparece dos veces si está presente en dos bases de datos distintas pues son tratadas como si fueran distintas. Esta decisión de diseño puede cambiar en la siguiente iteración para optar por considerar dichas variaciones iguales y establecer relaciones de uno a muchos con las tablas correspondientes.

```
CREATE VIEW application_view
AS  select
    s.dna_study_id,
    dnas.name as dna_study_name,
    cv.called_variation_id as variation_id,
    chr.chromosome_id,
    chr.name as chromosome_name,
    cv.chr_position as chromosome_position,
    cv.reference as reference_allele,
    cv.allele_a as alternative_allele_a,
    cv.allele_b as alternative_allele_b,
    wfi.sample_id as sample_id_db,
    s.id as sample_id,
    cs.clinical_significance_id,
    cs.name as clinical_significance,
    p.phenotype_id,
    p.name as phenotype_name,
    p.description as phenotype_description,
    si.stored_in_id as variation_id_in_data_source,
    ds.data_source_id,
    ds.name as data_source_name,
    g.gene_id as gene_id,
    g.name as gene_name,
    rp.research_publication_id as research_publication_id_db,
    rp.publication_site,
    rp.id as research_publication_id,
    rp.link as research_publication_link
from was_found_in wfi
left join sample s on s.sample_id = wfi.sample_id
left join dna_study dnas on dnas.dna_study_id = s.dna_study_id
left join called_variation cv on wfi.called_variation_id = cv.called_variation_id
left join called_curated_variation ccv on cv.called_variation_id = ccv.called_variation_id
left join chromosome chr on chr.chromosome_id = cv.chromosome_id
left join curated_variation cuv on cuv.curated_variation_id = ccv.curated_variation_id
left join clinical_significance cs on cs.clinical_significance_id = cuv.clinical_significance_id
left join predisposes_to pt on pt.curated_variation_id = cuv.curated_variation_id
left join phenotype p on p.phenotype_id = pt.phenotype_id
left join stored_in si on si.curated_variation_id = cuv.curated_variation_id
left join data_source ds on ds.data_source_id = si.data_source_id
left join is_located_in ili on ili.curated_variation_id = cuv.curated_variation_id
left join gene g on g.gene_id = ili.gene_id
left join supported_by sb on sb.curated_variation_id = cuv.curated_variation_id
left join research_publication rp on rp.research_publication_id = sb.research_publication_id
order by cv.called variation id;
```

Ilustración 23 script de creación de vista

5.2. Modelo de los servicios

Una vez definido el esquema relacional de la base de datos y la vista generada a partir de este para poder trabajar con dichos datos, se definen los requisitos del servicio. Para ello se han empleado tablas donde se explicita el nombre del requisito, una pequeña descripción y los parámetros de entrada y salida. Puesto que la aplicación está dividida en tres pantallas, también lo está nuestra especificación. De este modo, partimos de las especificaciones de la pantalla de selección de datos:

Nombre	Obtener bases de datos

Descripción	Obtener todas las bases de datos disponibles en el sistema	
Entrada	**Descripción**	**Tipo**
-	-	-
Salida	**Descripción**	**Tipo**
datasources	Conjunto de bases de datos presentes en el sistema	List<DataSource>

Nombre	Obtener estudios de ADN	
Descripción	Obtener todos los estudios de ADN existentes en el sistema	
Entrada	**Descripción**	**Tipo**
-	-	-
Salida	**Descripción**	**Tipo**
dnaStudies	Conjunto de estudios de ADN presentes en el sistema	List<DNAStudy>

Nombre	Añadir base de datos	
Descripción	Seleccionar una base datos para que su contenido sea utilizado en el proceso de recuperación de datos	
Entrada	**Descripción**	**Tipo**
Id	Identificador de la base de datos seleccionada	Número entero
Salida	**Descripción**	**Tipo**
-	-	-

Nombre	Añadir estudio de ADN	
Descripción	Seleccionar un estudio de ADN para que su contenido sea utilizado en el proceso de recuperación de datos	
Entrada	**Descripción**	**Tipo**
Id	Identificador del estudio de ADN seleccionado	Número entero
Salida	**Descripción**	**Tipo**
-	-	-

Nombre	Seleccionar muestra de estudio	
Descripción	Seleccionar tan solo una muestra perteneciente a un estudio de ADN (y no el estudio de ADN completo) para que su contenido sea utilizado en el proceso de recuperación de datos	
Entrada	**Descripción**	**Tipo**
Id	Identificador de la muestra seleccionada.	Número entero
Salida	**Descripción**	**Tipo**
-	-	-

Nombre	Deseleccionar base de datos	
Descripción	Excluye una base datos ya seleccionada para que su contenido no sea utilizado en el proceso de recuperación de datos	
Entrada	**Descripción**	**Tipo**
Id	Identificador de la base de datos seleccionada	Número entero
Salida	**Descripción**	**Tipo**
-	-	-

Nombre	Deseleccionar estudio de ADN	
Descripción	Excluye un estudio de ADN ya seleccionado para que su contenido sea utilizado en el proceso de recuperación de datos	
Entrada	**Descripción**	**Tipo**
Id	Identificador del estudio de ADN seleccionado	Número entero
Salida	**Descripción**	**Tipo**
-	-	-

Tras definir las especificaciones de la primera pantalla, pasamos a la segunda pantalla o pantalla de tratamiento y filtrado de datos:

Nombre	Obtener muestras por variaciones

Descripción	Se obtienen las muestras con el número de variantes de cada una para poder analizarlas y filtrarlas visualmente.	
Entrada	**Descripción**	**Tipo**
-	-	-
Salida	**Descripción**	**Tipo**
Samples	Conjunto de muestras recuperadas en base a la selección previa.	Map<Sample, Integer>

Nombre	Seleccionar muestra	
Descripción	Seleccionar una muestra de las disponibles que no lo esté para incluirla en el proceso de recuperación y filtrado de datos. Cuando se realiza esta acción, todos los widgets existentes deben actualizarse con la nueva información.	
Entrada	**Descripción**	**Tipo**
Id	Identificador de la muestra seleccionada.	Número entero
Salida[36]	**Descripción**	**Tipo**
chromosomeVsVariant	Colección de identificadores de cromosomas con el número de muestras que pertenecen a dicho cromosoma y superan todos los filtros.	Map<String, Integer>
phenotypeVsVariant	Colección de identificadores de fenotipos con el número de muestras que pertenecen a dicho fenotipo y superan todos los filtros.	Map<String, Integer>
clinicalSignificanceVsVariant	Colección de identificadores de significancias clínicas con el número de muestras que pertenecen a dicha significancia clínica y superan todos los filtros.	Map<String, Integer>
resultTable	Colección de muestras que cumplen todas las condiciones	List<ApplicationView>

Nombre	Obtener cromosomas por variaciones

[36] Esta salida es consecuencia de la ejecución del método actualizar. Para mayor comodidad cuando esta se repita este hecho la salida se cambiará por "actualizar()"

Descripción	Se obtienen los cromosomas que contienen al menos una variación que supera los filtros con el número de variaciones que posee.	
Entrada	**Descripción**	**Tipo**
-	-	-
Salida	**Descripción**	**Tipo**
chromosomeVsVariant	Conjunto de identificadores de cromosomas con el número de variaciones que contiene	Map<String, Integer>

Nombre	Obtener fenotipos por variaciones	
Descripción	Se obtienen los fenotipos que contienen al menos una variación que supera los filtros con el número de variaciones que posee.	
Entrada	**Descripción**	**Tipo**
-	-	-
Salida	**Descripción**	**Tipo**
phenotypeVsVariant	Conjunto de identificadores de fenotipos con el número de variaciones que contiene	Map<String, Integer>

Nombre	Obtener significancias clínicas por variaciones	
Descripción	Se obtienen las significancias clínicas que contienen al menos una variación que supera los filtros con el número de variaciones que posee.	
Entrada	**Descripción**	**Tipo**
-	-	-
Salida	**Descripción**	**Tipo**
clinicalSignificanceVsVariant	Conjunto de identificadores de significancias clinicas con el número de variaciones que contiene	Map<String, Integer>

Nombre	filtrar	
Descripción	Se filtra el conjunto de variaciones presentes a partir de uno de los elementos gráficos presentes en esta pantalla. Se puede filtrar por cromosoma, fenotipo, muestra y significancia clínica	
Entrada	**Descripción**	**Tipo**

filtro	Objeto que contiene el tipo de filtro a añadir y el identificador del valor del filtro	FilterChange
Salida	**Descripción**	**Tipo**
	actualizar()	

Nombre	Eliminar filtro
Descripción	Se elimina un filtro existente en el conjunto de filtros existentes a partir de la selección de uno de los filtros presentes en el widget de filtros activos.

Entrada	**Descripción**	**Tipo**
filtro	Objeto que contiene el tipo de filtro a añadir y el identificador del valor del filtro	FilterChange
Salida	**Descripción**	**Tipo**
	actualizar()	

Finalmente, en la tercera y última pantalla, también llamada de visualización de datos, se ha especificado la siguiente funcionalidad:

Nombre	Filtrar
Descripción	Se debe poder filtrar en la tabla de la pantalla de visualización de datos sin que esto modifique el estado de la pantalla de tratamiento y filtrado de datos mediante la utilización de un cuadro de búsqueda.

Entrada	**Descripción**	**Tipo**
-	-	-
Salida	**Descripción**	**Tipo**
-	-	-

5.3. Modelo de interfaz gráfica

Finalmente pasamos a describir el modelo seguido para la selección, tratamiento y filtrado de los datos disponibles. La aplicación se divide en tres pantallas o secciones que si bien serán descritas en detalle en el apartado de la interfaz gráfica, es conveniente mencionar, pues los servicios se han modelado de acuerdo a esta distribución. En primer lugar, se describe la primera pantalla, encargada de presentar al usuario de forma clara y sencilla los datos contenidos en la base de datos. En segundo lugar, la pantalla de tratamiento o filtrado de datos, esta pantalla va a contener la gran mayoría de las interacciones por parte del usuario ya que proporciona las herramientas necesarias para realizar el análisis las cuales serán desarrolladas en su correspondiente sección. En tercer y último lugar se haya la pantalla encargada de mostrar el resultado del tratamiento realizado en la segunda pantalla, esto es, un listado con los datos relevantes de aquellas variaciones seleccionadas en la primera pantalla que cumplan los requisitos explicitados en la segunda pantalla. Cabe señalar que, debido a la naturaleza colaborativa del caso de estudio, estos servicios deben ser potencialmente escalables, de modo que en la *GUI* se deben poder disponer de tantas ocurrencias de pantallas o widgets como se deseen y el servidor debe poder dar soporte a la potencial escalabilidad necesaria para el uso simultáneo de un elevado número de pantallas ya sea en diferentes dispositivos o en el mismo.

5.3.1. Selección de datos

Esta primera pantalla es la responsable de ofrecer al usuario una vista de los posibles datos con los que puede trabajar. Está compuesta de dos listados:

- Listado de los análisis de ADN: Listado de los estudios realizados a pacientes almacenados en la base de datos. Por cada análisis se crea un nuevo elemento en la lista que a su vez se subdivide en dos: El primero de ellos se compone del nombre del estudio seguido del número de muestras contenido en dicho estudio. El segundo está formado por una lista de las muestras pertenecientes a dicho análisis incluyendo su nombre e identificador y el número de variaciones que contiene cada muestra. El usuario puede seleccionar tanto estudios enteros como muestras sueltas de dichos estudios
- Listado de las bases de datos: Listado de las bases de datos cuyo contenido ha sido adaptado al modelo con el que se trabaja en la aplicación y explicado

anteriormente. Cada elemento de la lista está formado por el nombre que identifica a la base de datos y el número de variaciones que contiene.

Al iniciar la aplicación, al usuario se le ofrecerán estas dos listas, de las que elegirá libremente los análisis o muestras que desea investigar y las bases de datos con las que contrastar las variaciones seleccionadas en la primera lista. Una vez ha seleccionado los datos estos son enviados al servidor que recupera las variaciones correspondientes y actualiza automáticamente las demás pantallas poblándolas con los datos cargados desde el servidor. En cualquier momento pueden ser añadidos nuevos datos desde esta pantalla y es responsabilidad del servidor mantener el estado actual y añadirle los nuevos datos, de modo que el nuevo estado sea igual que el anterior, pero realizando el filtrado al nuevo conjunto de datos, resultante del conjunto anterior más los nuevos datos recién añadidos. La carga de las variaciones de las muestras seleccionadas debe limitarse a las variaciones que estén presentes en alguna de las bases de datos seleccionadas o, en su defecto, en ninguna de ellas.

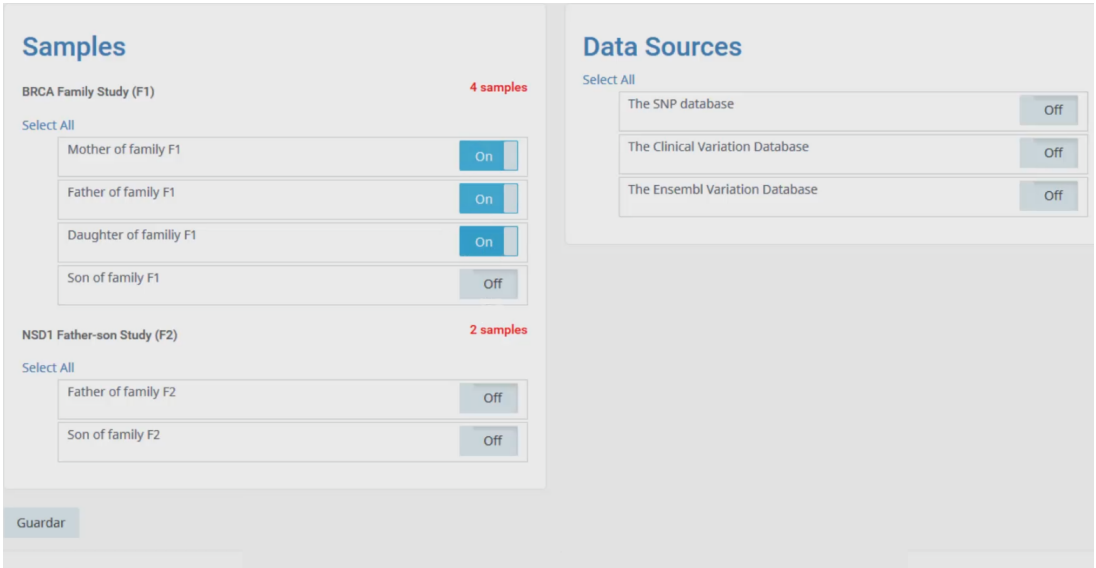


Ilustración 24 Selección de datos

5.3.2. Tratamiento y filtrado de datos

Vamos a analizar ahora la segunda pantalla; tras seleccionar las variaciones en la primera pantalla, los widgets que van a ser utilizados para el filtrado de datos se actualizan automáticamente mostrando las variaciones y sus correspondientes agrupaciones sin aplicársele ningún filtro. Esta pantalla dispone de una gran

versatilidad ya que le pueden añadir y quitar todos los widgets que deseemos de manera transparente para el usuario.

Se disponen de 5 widgets distinto que pueden ser añadidos o eliminados, estos widgets vienen pre configurados para proporcionar al usuario un inicio rápido aunque, si fuera necesario, es posible modificar el formato de los gráficos. Si bien cada uno agrupa las variaciones de una manera distinta, todos comparten ciertas características comunes: Se pueden añadir o eliminar en cualquier momento, están interconectados (los datos visualizados dependen de sus propios filtros y de los filtros de todos los demás) y actualización instantánea ante cualquier modificación. Hablaremos primero de los widgets que pueden añadir filtros al conjunto de datos:

- Widget 1: Tabla muestra y variante: Se trata de una tabla que contiene todas las muestras seleccionadas para nuestra investigación. Cada elemento de la tabla consta del nombre de la muestra y el número total de muestras pertenecientes a dicha muestra; posee, además, un elemento visual de tipo *checkbox* para localizar fácilmente las muestras que están seleccionadas y las que no. Inicialmente están todas seleccionadas pero mediante un clic en el nombre de la muestra se puede deseleccionar y excluir las variaciones pertenecientes a dicha muestra

Ilustración 25 Tabla muestra y variante

- Widget 2: Gráfico cromosoma y variante: Dispone un gráfico de barras donde el eje horizontal muestra los cromosomas que contienen al menos

una variación de las presentes en el conjunto de datos a estudiar siendo el eje vertical el número total de variaciones que posee cada cromosoma del conjunto de datos seleccionados. Cuando se hace clic en la representación visual de un cromosoma (Su barra) las muestras se filtran y solo se muestran las variaciones anteriores que cumplan el nuevo criterio; de modo que pertenezcan todas al cromosoma seleccionado. Es posible seleccionar más de un cromosoma de modo que la visualización contendrá a las variaciones que cumplan los criterios anteriores y pertenezcan al conjunto de cromosomas seleccionados. La selección de un cromosoma implica que en la tabla de filtros se cree una nueva entrada simbolizando que se ha añadido un filtro con dicho cromosoma. En el

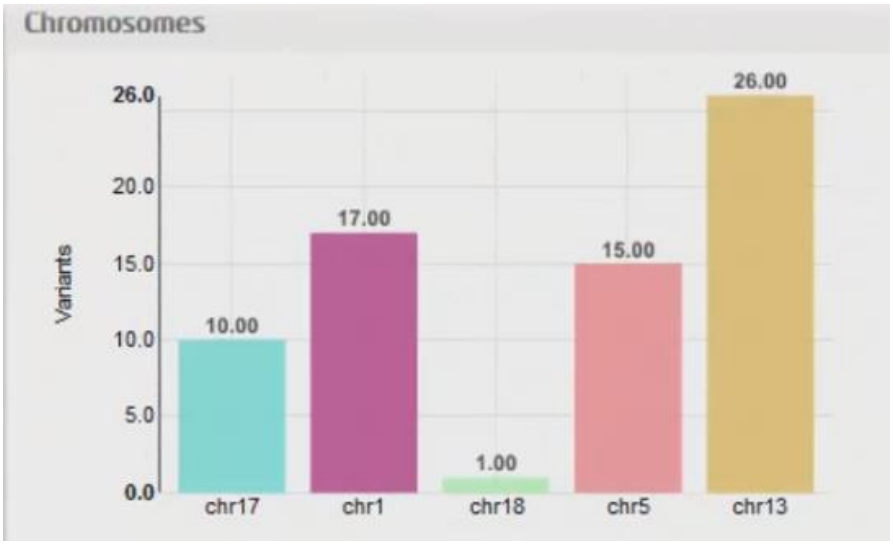


Ilustración 26 Gráfico cromosoma y variante

caso de que hayan varios cromosomas seleccionados se creará una entrada por cada cromosoma seleccionado.

- Widget 3: Gráfico fenotipo y variante: Gráfico de circular o de tipo tarta, que muestra los fenotipos presentes en el conjunto de variaciones seleccionadas para el estudio. El espacio del gráfico se divide según el número de variaciones de cada fenotipo. En cada sección aparece el número total de variaciones así como el color asignado en la leyenda para dicho fenotipo; del mismo modo, cuando se pasa el ratón por encima de uno de los elementos aparece un mensaje con el fenotipo de dicha

fracción del gráfico. Cuando se hace clic en la representación visual de un fenotipo las muestras se filtran de nuevo siendo tan solo visibles las variaciones anteriores que predispongan al fenotipo seleccionado. La selección de un fenotipo implica que en la tabla de filtros se cree una nueva entrada simbolizando que se ha añadido un filtro con dicho fenotipo.

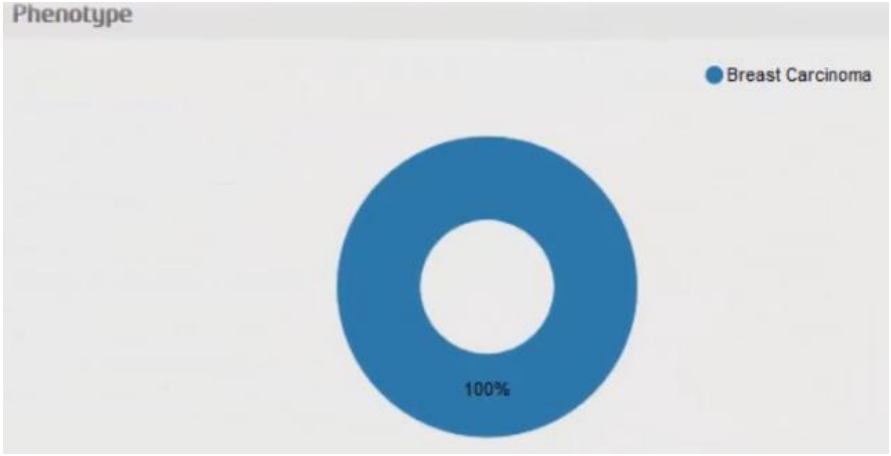


Ilustración 27 Gráfico fenotipo y variante

- Widget 4: Gráfico significancia clínica y variante: Dispone un gráfico de barras donde el eje horizontal muestra las distintas significancias cínicas que contienen las variaciones que se hayan presentes en el conjunto de datos a estudiar siendo el eje vertical el número total de variaciones que presentan dicha significancia clínica del conjunto de datos seleccionados. Cuando se hace clic en la representación visual de una significancia clínica (Su barra) las muestras se filtran de nuevo siendo tan solo visibles las variaciones anteriores que posean la significancia clínica seleccionada. Es posible seleccionar más de una significancia clínica de modo que la visualización contendrá a las variaciones que cumplan los criterios anteriores y pertenezcan al conjunto de significancias clínicas seleccionados. La selección de una significancia clínica implica que en la tabla de filtros se cree una nueva entrada simbolizando que se ha añadido un filtro con dicha significancia clínica. En el caso de que hayan varias significancias clínicas seleccionadas se creará una entrada por cada significancia clínica seleccionada.

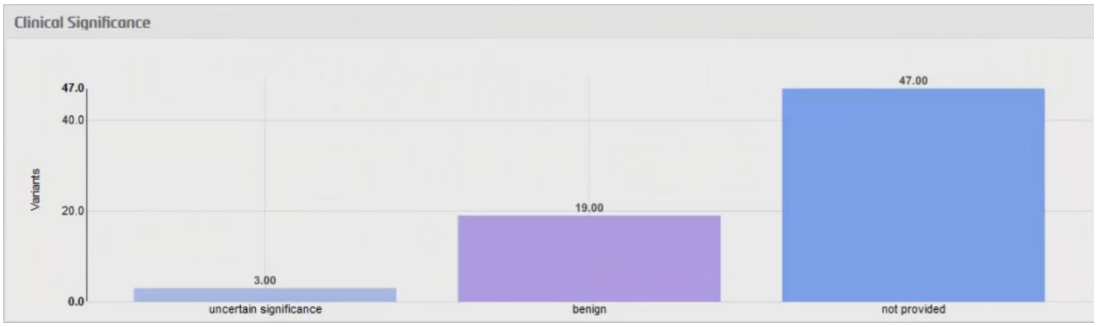


Ilustración 28 Gráfico significancia clínica y variante

Estos gráficos componen la parte más visual de esta pantalla. Cuando un filtro es añadido, automáticamente todos los elementos de todas las segundas pantallas y todas las terceras pantallas conectadas al servidor son actualizados, presentando el nuevo conjunto de datos. Cuando se aplican una serie de criterios que hace que uno de los elementos de los gráficos devuelva el valor 0, este sigue apareciendo con dicho valor de modo que el usuario es consciente de que ha hecho una selección que implica la exclusión completa de esa característica. En el caso de que se llegue a un conjunto de datos que devuelvan el conjunto vacío, los gráficos se presentan vacíos alertando al usuario de que no hay ninguna variación que cumpla los requisitos que ha utilizado.

Por último, el último widget disponible es la tabla de filtros. Esta tabla contiene todos los filtros aplicados (a excepción de las muestras seleccionadas, que tienen su propio elemento visual de identificación); cada elemento de la tabla consta del tipo de filtro seleccionado con de su valor (Por ejemplo, Cromosoma 3) y un botón de borrado, constituye un punto único para que el usuario pueda observar rápidamente los filtros seleccionados y, si lo desea, eliminarlos. Cuando se desea eliminar un filtro tan solo hay que seleccionar en borrar en el elemento que se desee eliminar aunque también existe la opción de eliminarlos todos. Puesto que son filtros no encadenados (independientes) no hay ninguna restricción en el orden que se desean borrar.

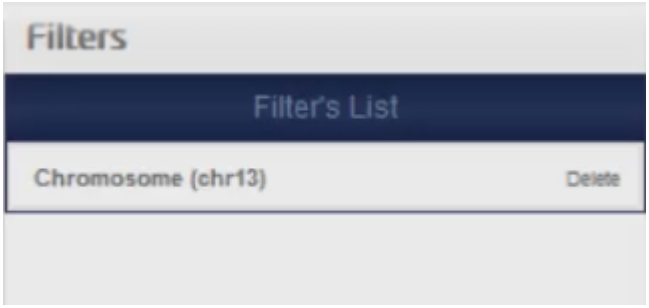
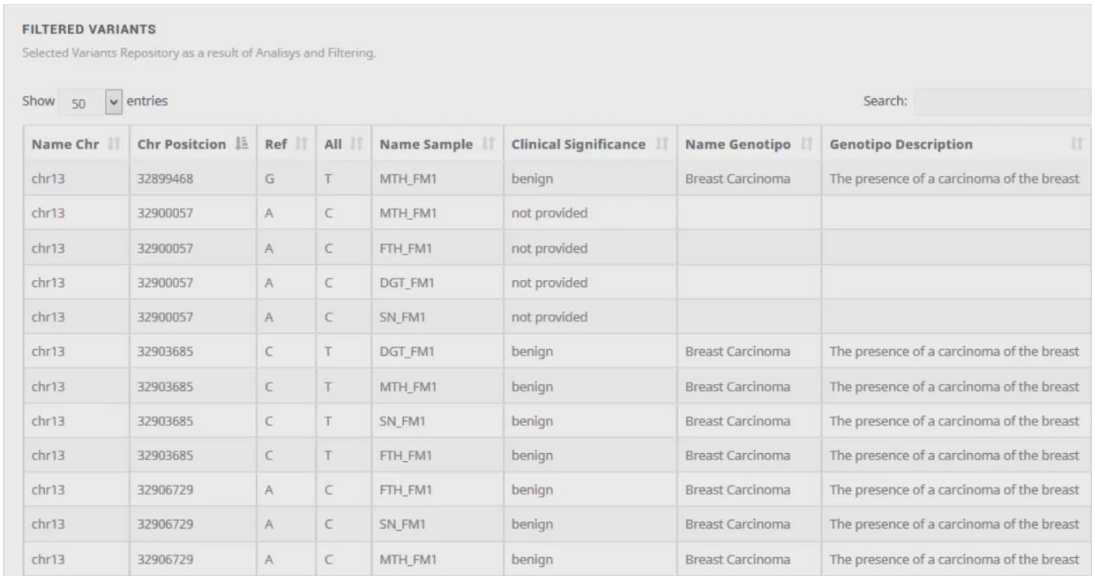


Ilustración 29 Tabla de filtros

5.3.3. Resultado

La tercera y última pantalla, contiene una tabla ordenable con todas las variaciones que cumplen los criterios establecidos. Contiene una fila por cada variación y cada fila contiene las siguientes columnas: Cromosoma al que pertenece, posición en el cromosoma, alelos de referencia y alternativo, nombre de la muestra, significancia clínica y fenotipo. Para próximas versiones se modificarán los campos para que presente las siguientes columnas: nombre del estudio al que pertenece, identificación de la variación, nombre del cromosoma donde se ubica la variación, posición de la variación dentro de dicho cromosoma, alelo de referencia, alelo alternativo a, alelo alternativo b (se trata de los alelos que contiene dicha variación, se almacenan los dos para cubrir posibles casos de heterozigosis), significancia clínica, muestra a la que pertenece, fenotipo con el que está relacionado, base de datos donde se encuentra dicha variación y gen al que pertenece.



FILTERED VARIANTS

Selected Variants Repository as a result of Analisys and Filtering.

Show 50 entries Search:

Name Chr	Chr Positcion	Ref	All	Name Sample	Clinical Significance	Name Genotipo	Genotipo Description
chr13	32899468	G	T	MTH_FM1	benign	Breast Carcinoma	The presence of a carcinoma of the breast
chr13	32900057	A	C	MTH_FM1	not provided		
chr13	32900057	A	C	FTH_FM1	not provided		
chr13	32900057	A	C	DGT_FM1	not provided		
chr13	32900057	A	C	SN_FM1	not provided		
chr13	32903685	C	T	DGT_FM1	benign	Breast Carcinoma	The presence of a carcinoma of the breast
chr13	32903685	C	T	MTH_FM1	benign	Breast Carcinoma	The presence of a carcinoma of the breast
chr13	32903685	C	T	SN_FM1	benign	Breast Carcinoma	The presence of a carcinoma of the breast
chr13	32903685	C	T	FTH_FM1	benign	Breast Carcinoma	The presence of a carcinoma of the breast
chr13	32906729	A	C	FTH_FM1	benign	Breast Carcinoma	The presence of a carcinoma of the breast
chr13	32906729	A	C	SN_FM1	benign	Breast Carcinoma	The presence of a carcinoma of the breast
chr13	32906729	A	C	MTH_FM1	benign	Breast Carcinoma	The presence of a carcinoma of the breast

Ilustración 30 Tabla de variaciones

Como hemos visto anteriormente en el apartado vista del modelo de la base de datos, pueden haber campos vacíos dependiendo de si se encuentra o no una variación en alguna base de datos externa. La tabla proporciona mecanismos para ordenar las variaciones por cualquiera de los criterios anteriormente citados además de ofrecer la posibilidad de búsqueda utilizando como criterio de búsqueda cualquier campo; ofrece además soporte a búsquedas parciales.

6. Implementación

En esta sección se explica la implementación seguida en base a la arquitectura definida. La implementación ha seguido en el modelo de la ilustración Ilustración 18 Arquitectura. Desplegando la interfaz gráfica de usuario y la base de datos PostgreSQL en un servidor dentro de Fiware y la lógica de la aplicación en un servidor Apache Tomcat externo a Fiware. A continuación se enumeran y explican brevemente las clases implementadas para la lógica de la aplicación.

Cabe destacar q que en el diagrama de clases, se han obviado las clases resultado de mapear el modelo de la base de datos (pues sigue la misma estructura que el presentado en la sección 5.1) y los constructores y métodos getters y setters de las clases presentes en el diagrama. La aplicación está divida en paquetes según su funcionalidad. Así, podemos encontrar cinco paquetes distintos:

- com.gemdomus.mappedClasses: Este paquete contiene todas las clases que han sido mapeadas mediante Hibernate Tools así como la clase mapeada manualmente para la vista generada en la base de datos.
- com.gemdomus.returnedClasses: Almacena las dos clases utilizadas en la lógica de negocio para proporcionar la información de la pantalla de selección de datos.
- com.gemdomus.controllers: Contiene los controladores utilizados para interactuar con la base de datos. Existen tres controladores: Uno para obtener los estudios de ADN, otro para obtener las bases de datos disponibles y, finalmente, uno para obtener las variaciones.
- com.gemdomus.dao: Contiene las clases encargadas de comunicarse directamente con la base de datos (*Data Access Object*). Proveen a los controladores.
- com.gemdomus.internalClasses: En este paquete están presentes todas las clases que intervienen en la lógica del servicio.

- com.gemdomus.websocket: Contiene el Websocket utilizado para la comunicación cliente-servidor y los archivos de configuración necesarios para su correcto mapeo y funcionamiento.

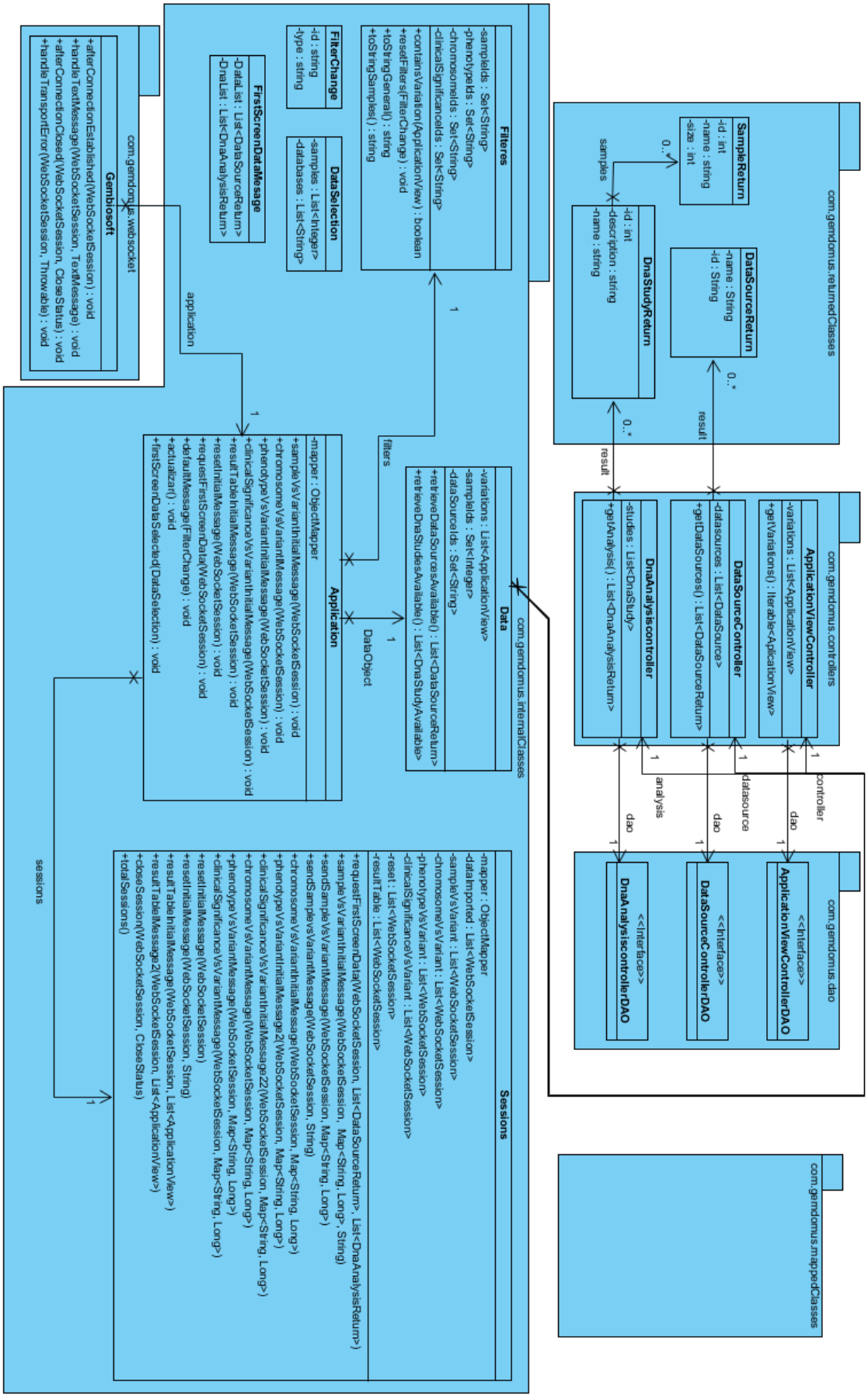


Ilustración 31 diagrama de clases de la aplicación

Tras enumerar los paquetes en que se divide la lógica del servicio, se entra en el detalle de cada una de las clases partiendo de la clase Application, que ejerce de clase central.

La clase Application contiene una instancia de las clases Filter, Data y Sessions (descritas psoteriormente). Actúa de clase central la cual gestiona toda la información entrante y decide, en consecuencia, que clase debe o no intervenir. Posee los siguientes métodos:

- sampleVsVariantInitialMessage: Cuando un gráfico que deba presentar las variaciones agrupadas por la muestra a la que pertenecen se conecte por primera vez, este método es llamado. En él se almacena la sesión y se le envía la información necesaria.
- chromosomeVsVariantInitialMessage: Cuando un widget que deba presentar las variaciones agrupadas por el cromosoma al que pertenecen se conecte por primera vez, este método es llamado. En él se almacena la sesión y se le envía la información necesaria.
- phenotypeVsVariantInitialMessage: Cuando un widget que deba presentar las variaciones agrupadas por el fenotipo al que pertenecen se conecte por primera vez, este método es llamado. En él se almacena la sesión y se le envía la información necesaria.
- clinicalSignificanceVsVariantInitialMessage; Cuando un widget que deba presentar las variaciones agrupadas por la significancia clínica que poseen se conecte por primera vez, este método es llamado. En él se almacena la sesión y se le envía la información necesaria.
- resultTableInitialMessage: Cuando un widget que deba presentar las variaciones siguiendo el estilo de la pantalla de presentación de resultados se conecte por primera vez, este método es llamado. En él se almacena la sesión y se le envía la información necesaria.
- resetInitialMessage: Cuando un widget que deba mostrar los filtros presenten en la actual selección de datos se conecte por primera vez, este método es llamado. En él se almacena la sesión y se le envía la información necesaria.
- requestFirstScreenData: Cuando la pantalla de selección de datos se conecta por primera vez, necesita recibir los estudios de ADN y las bases de datos disponibles para que el usuario pueda hacer su propia selección.

Este método se encarga de almacenar la sesión y proporcionar dicha información mediante el envío de un mensaje con el formato de la clase FirstScreenDataMessage.

- defaultMessage: Cuando uno de los widgets de la pantalla de filtrado de datos envía información (añadir un filtro), este método es llamado. Recibe un mensaje con formato FilterChange y delega en la clase Filters la gestión de este nuevo criterio a la lista de filtros ya existentes.
- actualizar: Ante la modificación de un criterio en el conjunto de filtros existentes, este método es llamado para actualizar la información de los widgets que así lo requieran.
- firstScreenDataSelected: Si se recibe un mensaje que sigue el formato de la clase interna DataSelection, se ejecuta este método para almacenar los filtros en función de la selección realizada por el usuario en la pantalla de selección de datos. Además, carga las variaciones que cumplan los criterios en la base de datos.

La clase Filters se encarga de la gestión de los filtros activos durante la ejecución de la aplicación. Contiene tres colecciones, una por cada criterio que se puede aplicar a un filtro: Muestra a la que pertenece una variación, cromosoma al que pertenece una variación, fenotipo provocado por una variación y significancia clínica de una variación. Sus principales métodos son: resetFilters que elimina un filtro del conjunto de filtros activos; Para ello, recibe como parámetro un objeto de tipo FilterChange y obtiene el identificador del filtro que debe borrar del valor del atributo type de dicho objeto. containsVariation recibe una variación como parámetro de entrada y da como salida si dicha variación cumple con los filtros activos en ese momento. Finalmente, el método toString se ha divido en dos, uno incluye los filtros activos referentes a las muestras y el otro todos los demás filtros. Esto es debido a que si bien los filtros referentes a cromosoma, fenotipo y significancia clínica están presentes en un wiget dedicado a los filtros activos, los filtros referentes a las muestras se encuentran en el widget que presenta las variaciones agrupadas por la muestra a la que pertenecen, ya que el número de muestras seleccionadas puede ser potencialmente mayor que el resto de criterios de filtrado y tenderán a estar seleccionadas.

La clase Sessions es responsable del almacenamiento de las sesiones, así como de su ciclo de vida y el envío de los mensajes a estas. Contiene siete colecciones para almacenar las

sesiones entrantes (Una por cada tipo de widget que puede ser conectado al servidor). Se han definido una serie de métodos por cada widget:

- Geter y setter: Métodos para obtener y asignar el valor de cada colección de sesiones.
- InitialMessage: Método ejecutado la primera vez que un widget se conecta al servidor. Almacena la sesión y le envía la información correspondiente.
- Message: Método opcional ejecutado cuando se envía un mensaje a un widget ya conectado producto de una actualización en el conjunto de filtros activos.

Finalmente, contiene un método encargado de eliminar una sesión de las colecciones internas cuando esta es cerrada.

Una serie de clases son utilizadas para la comunicación entre el cliente y el servidor. Para ello son serializadas y deserializadas a JSON utilizando la librería Jackson. Existen tres de estas clases:

- FilterChange: Esta clase se utiliza cuando se añade o elimina un filtro debido a una interacción del usuario con la interfaz de usuario. Está compuesta de dos campos: id y type.
- FirstScreenDataMessage: Clase interna utilizada para el envío de la información a la pantalla de selección de datos cuando se conecta por primera vez. Está compuesta de dos atributos: Una colección de DataSourceReturn (DataList) y una colección de DnaAnalysisReturn (DnaList).
- DataSelection: Clase empleada para comunicar al servidor la configuración escogida por el usuario en la pantalla de selección de datos. Está compuesta de dos atributos: Una lista de enteros para indicar los identificadores de las muestras (samples) y una lista de cadenas de texto para indicar lo identificadores de la base de datos (databases).

La clase Data posee tres colecciones: Una lista que contiene las variaciones obtenidas de la base de datos, el conjunto de identificadores de las bases de datos seleccionadas y el conjunto de identificadores de las muestras seleccionadas. Posee dos métodos: retireveDataSourcesAvailable devuelve todas las bases de datos disponibles presente en la base de datos siguiendo el formato de la clase DataSourceReturn; retrieveDNAStudiesAvailable devuelve todos los estudios de ADN disponibles presentes en

la base de datos siguiendo el formato de la clase DNAStudyReturn. Esta clase contiene tres controladores para poder cumplir con su funcionalidad.

Se han definido tres controladores para interactuar con la base de datos. El primero de ellos es la clase ApplicationViewController, contiene un atributo para almacenar las variaciones recuperadas y un único método: getVariations que obtiene, a partir del DAO, la lista de variaciones de la base de datos. DataSourceController y DnaAnalysisController proveen de las bases de datos y análisis de ADN respectivamente; poseen la misma estructura que el controlador anterior, con un único método get y un DAO. Su diferencia radica en que los datos devueltos no siguen el formato de la base de datos sino un nuevo formato: Para DataSourceController el método getDataSources devuelve una lista de DataSourceReturn, esta clase contiene dos campos de cadenas de texto (id y name), Para DnaAnalysisController el método getDnaAnalysis devuelve una lista de DnaAnalysisReturn, esta contiene los siguientes campos: id (entero), desription (cadena de texto), name (cadena de texto) y samples (Lista de SampleReturn).

Cada DAO esta implementado como una interfaz que extiende de la interfaz de Spring CrudRepository. Se ha parametrizado con la clase JAVA anotada que mapea y el tipo de su identificador. Estas interfaces están anotadas con @Repository para indicar que se trata de un DAO. Al extender de la interfaz CrudRepository y estar parametrizada, Spring es capaz de proporcionar de manera transparente funcionalidad CRUD (insertar, eliminar, modificar y obtener elementos).

La clase Gembiosoft implementa la funcionalidad de un WebSocket, modificando los métodos básicos. Contiene una instancia de la clase Application y se encarga de derivar en esta los datos que recibe desde las distintas sesiones.

Para finalizar, se adjunta una tabla con todas las dependencias empleadas en el desarrollo de los servicios:

Group id	Artifact id	Version
Com.fasterxml.jackson.core	Jackson-core	2.6.5
java.servelt	java.servlet-api	3.1.0
org.hibernate	Hibernate-entitymanager	4.3.11.FINAL
org.postresql	Postgresql	9.4.1208.jre7
org.springframework.boot	Spring-boot-maven-plugin	1.3.3.RELEASE
org.springframework.boot	Spring-boot-starter-data-jpa	1.3.3.RELEASE

org.springframework.boot	Spring-boot-starter-data-rest	1.3.3.RELEASE
org.springframework.boot	spring-boot-starter-test	1.3.3.RELEASE
org.springframework.boot	Spring-boot-starter-tomcat	1.3.3.RELEASE
org.springframework.boot	spring-boot-starter-web	1.3.3.RELEASE
org.springframework.boot	spring-boot-starter-websocket	1.3.3.RELEASE
org.springframework.data	Spring-data-jpa	1.9.4.RELEASE

Tabla 3 Dependencias utilizadas

6.1. Comunicación con la capa de persistencia

El acceso a los datos resulta de vital importancia; la base de datos utilizada emplea tecnología PostgresSQL. En primer lugar, y aprovechando las facilidades que provee la herramienta de gestión de dependencias Maven, se ha incorporado la dependencia necesaria para añadir el driver *ODBC* (*open database connectivity*); para ello se ha añadido al archivo pom.xml su dependencia (groupId: "org.postgresql", artifactId: "postresql"). Para configurar JPA en Spring mediante xml es necesario realizar los siguientes pasos: Configurar el Bean de la base de datos, configurar el bean entityManagerFactoryBean, configurar del Bean transactionManager, permitir la configuración mediante anotaciones y configurar JPA para Spring Data.

El siguiente paso ha consistido en definir el archivo persistence.xml, se trata de un archivo de configuración que forma parte de la especificación de JPA. Este archivo contiene uno o más Beans. Esta unidad de persistencia es inyectada posteriormente en la configuración del Bean LocalEntityManagerBean. En este archivo ha definido una unidad de persistencia con la configuración para acceder a la base de datos utiliza como provider: hrg.hibernate.jpa.HibernatePersistenceProvider, a este provider se le añaden las propiedades necesarias para establecer una conexión con la base de datos; dichas propiedades son las siguientes:

- hibernate.connection.driver_class: indica el driver que va a ser utilizado para establecer la conexión. En este caso el driver de PostgreSQL (org.postgresql.Driver)
- hibernate.connection.username: Usuario con el que se va a acceder a la base de datos. Cada usuario puede tener un rol y unos privilegios diferentes.

- hibernate.connection.password: Contraseña empleada para acceder con el usuario definido anteriormente
- hibernate.connection.url: dirección utilizada para establecer la conexión entre la aplicación y la base de datos. Sigue el patrón "jdbc:postresql://ip:5432/nombreBaseDatos".
- hibernate.dialect: Se trata de una propiedad opcional ya que Hibernate es capaz de inferir el lenguaje a partir de una lista interna de lenguajes a los que Hibernate da soporte, aunque es recomendable definirlo. Se le asigna el valor org.hibernate.dialect.PostgreSQLDialect.

Una vez definido el archivo persistence.xml, ya se ha definido el Bean de la base de datos. Esta información debe ser añadida a la configuración de la aplicación. Las aplicaciones de Spring poseen un archivo xml llamado applicationContext.xml, este es el encargado de gestionar los Beans de la capa de persistencia y de la lógica de negocio. En dicho archivo se configuran los dos Beans restantes:

- EntityManagerFactory: Se encarga de gestionar las entidades definidas en la aplicación, se trata de un Bean de la clase "org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean". Es configurado mediante las propiedades persistenceUnitName, este Bean encapsula el Bean definido en el archivo persistence.xml y contiene todos los datos relativos a la conexión con la unidad de persistencia. JpaVendorAdapter contiene un Bean de tipo "org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter". Finalmente, se define una serie de propiedades relativas a JPA: hibernate.dialect para definir el lenguaje de consulta; hibernate.show_sql puede tomar valor verdadero o falso e indica si se desea registrar en el log las consutlas sql generadas automáticamente por Hibernate; hibernate.format_sql señala si la consulta sql se refactoriza para hacerla más entendible a cambio de ocupar más espacio, su valor puede ser verdadero o falso; finalmente, la propiedad hibernate.hbm2ddl.auto valida o exporta automáticamente el esquema cuando se crea el Bean SessionFactory, puede tomar los valores validate (valida el esquema sin hacer cambios en la base de datos), update (actualiza el esquema), create

(vuelve a crear el esquema, destruyendo los datos almacenados anteriormente) o create-drop (borra el esquema al finalizar la sesión)

- TransactionManager*:* se trata de un Bean de tipo "org.springframwork.orm.jpa.jpaTransactionManager", se encarga de gestionar las transacciones a la base de datos y recibe como propiedad la entityManagerFactory configurada anteriormente.

Finalmente se configura JPA para Spring Data, para ello, basta con añadir la propiedad "jpa:repositories base-package" y darle como valor el directorio que contiene los repositorios de las interfaces. Spring boot permite el uso de anotaciones para declarar *beans* y que estos sean escaneados automáticamente; para ello, a la propiedad "context:component-scan base-package" se le dará el valor del paquete que contiene los componentes.

Hibernate ofrece herramientas para, haciendo uso de una conexión a la base de datos con los parámetros definidos, mapear automáticamente el modelo de la base de datos, obteniendo las clases con sus relaciones y mapeadas mediante anotaciones. En dichas anotaciones Se identifica cada clase como una Entity, esto indica que la clase es una entidad persistente. Esta y otras anotaciones ayudan a mapear el modelo y obtener su equivalente en JAVA.

Hibernate Tools no mapea las vistas automáticamente, de modo que la vista creada con todos los atributos utilizada para extraer los datos ha tenido que ser mapeada manualmente.

Anotación	Descripción	Atributos
@Table	Indica la tabla que mapea una clase	name
		uniqueConstaints
		schema
@Id	Especifica la clave primaria de una entidad	
@Column	Relaciona una columna con una variable	Nombre
		Nullable
		Precisión
@ManyToOne	Define una relación donde una variable única posee multiplicidad muchos a uno	fetch
		optional
		targetEntity
@JoinColumn	Especifica la columna utilizada para añadir las entidades a la colección de elementos	name
		nullable
		unique
		table
@OnteToMany	Define una asociación múltiple (colección) con multiplicidad uno a muchos	cascade
		mappedBy
		fetch
		targetEntity
@OneToOne	Define una asociación donde una variable unitaria posee una multiplicidad uno a uno	cascade
		Fetch
		mappedBy
		targetEntity
@ManyToMany	Define una asociación múltiple (colección) con multiplicidad muchos a muchos	cascade
		fetch
		mappedBy
@JoinTable	Utilizado en el mapeo de asociaciones, se especifica en el lado que posee la asociación	inverseJoinColumns
		joinColumns
		name
		uniqueConstraints

Tabla 4 Anotaciones utilizadas por Hibernate Tools

Spring, una vez más, combinado con Hibernate proporciona una capa de abstracción extra para la recuperación de los datos llamada Spring Data Repositories. Dicha capa tiene por objetivo reducir en gran medida la cantidad de código necesario para implementar el acceso a la capa de persistencia. Ofrece una interfaz llamada CrudRepository que proporciona funcionalidad para insertar, eliminar, obtener y modificar datos (CRUD), actúa utilizando el patrón marker interface[37], lo que permite una gran flexibilidad y simplicidad en el desarrollo de los objetos de acceso a la capa de persistencia. Recibe la clase sobre la que va a realizar consultas y el tipo del identificador

[37] Patrón de diseño usado con lenguajes que proveen información sobre el tipo de los objetos en tiempo de ejecución

como argumentos; de esta manera es capaz de generar automáticamente la funcionalidad CRUD transaccional [38] para dicha clase. Estas interfaces deben anotarse con @Repository para indicar que la clase actúa como un repositorio, es decir, como un mecanismo para encapsular el almacenamiento, recuperación y comportamiento de búsqueda emulando una colección de objetos [55] y se han utilizado para crear los objetos DAO[39] (uno por cada clase mapeada por Hibernate Tools) necesarios.

Estos objetos DAO han sido encapsulados en sus respectivos controladores. Cada controlador posee un contexto, instancia de la clase "ClassPathXmlApplicationContext" que se genera a partir del documento applicationContext.xml, y un DAO obtenido como un Bean del contexto. Se han definido tres controladores:

- Controlador de bases de datos: Contiene un único método que recupera el conjunto de las bases de datos disponibles y las devuelve con el formato adecuado. Para esto, dispone de dos colecciones, una para almacenar los elementos recuperados de la base de datos de tipo DataSource que actúa como el objeto de transferencia desde la base de datos, y otra colección con datos de tipo DataSourceReturn que actúa como objeto de negocio.
- Controlador de estudios de ADN: Contiene un único método que recupera el conjunto de los estudios de ADN disponibles para su estudio y los devuelve con el formato adecuado. Para esto, dispone de dos colecciones, una para almacenar los elementos recuperados de la base de datos de tipo DnaAnalysis que actúa como el objeto de transferencia desde la base de datos, y otra colección con datos de tipo DnaAnalysisReturn que actúa como objeto de negocio.
- Controlador de la vista creada: Contiene un único método que recupera el conjunto de las variaciones presentes en la vista creada. Dispone de una única colección de tipo ApplicationView ya que el objeto de transferencia coincide con el objeto de negocio.

Como se ha podido observar en la ilustración Ilustración 31 diagrama de clases de la aplicación, estos controladores están contenidos en la clase Data, esta es la encargada de gestionar tanto el acceso a la base de datos como la administración de los mismos. Cuando en la primera pantalla se seleccionan los datos con los que se va a trabajar, esta

[38] Una transacción es un conjunto de órdenes que se ejecutan formando una unidad de trabajo indivisible o atómica
[39] *Data Access Object*, objeto utilizado para acceder a los datos almacenados en sistemas externos

configuración es almacenada y las variaciones que cumplen dichos requisitos son almacenadas en memoria en una colección de variaciones cuyos objetos son de tipo ApplicationView.

Tan pronto como llega un mensaje identificando la conexión de la primera pantalla (el contenido del mensaje es descrito en el apartado API), y por tanto se requiere obtener los datos disponibles para su selección, el sistema inicia las llamadas necesarias para obtener dichos datos; para ello, y mediante el uso de los controladores, se recuperan los datos relativos las bases de datos disponibles y los estudios de ADN presentes. A continuación, la sesión es almacenada y se le envía la información obtenida por parte de los controladores; dicha información es presentada según se ha explicitado en el apartado de interfaz gráfica.

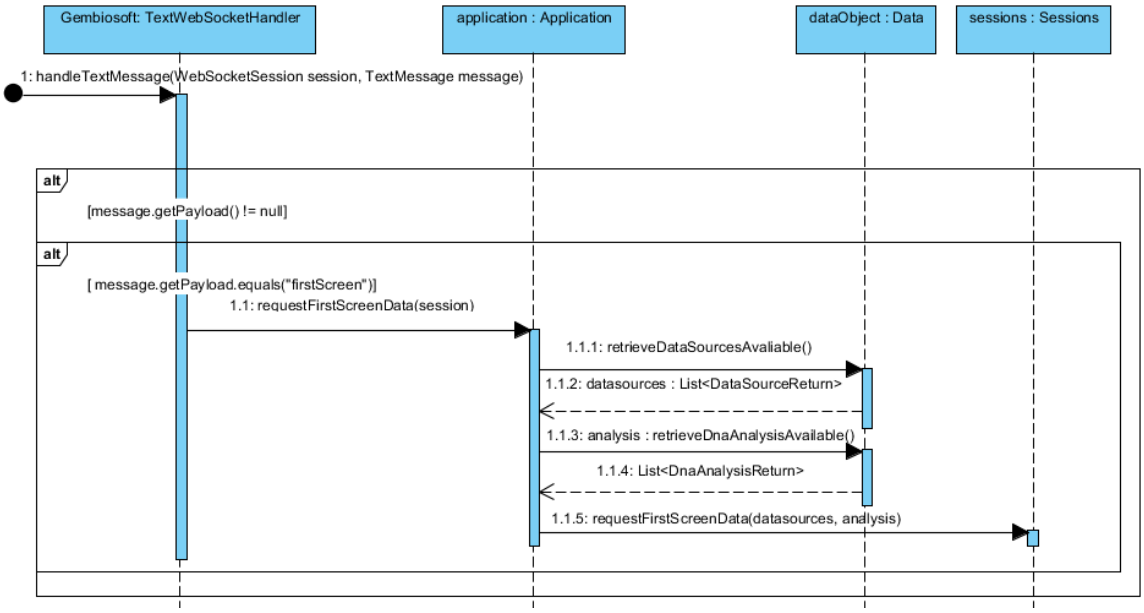


Ilustración 32 Mensaje inicial primera pantalla

El primer paso consiste en la obtención de las bases de datos disponibles. Para ello, como se ve en la ilustración Ilustración 33 el controlador presente en el objeto DataObject se sirve del dao para obtener todas las bases de datos disponibles para ser utilizadas en el análisis. Cuando las recibe, transforma los objetos recibidos a objetos de negocio los cuales son retornados hasta el objeto application.

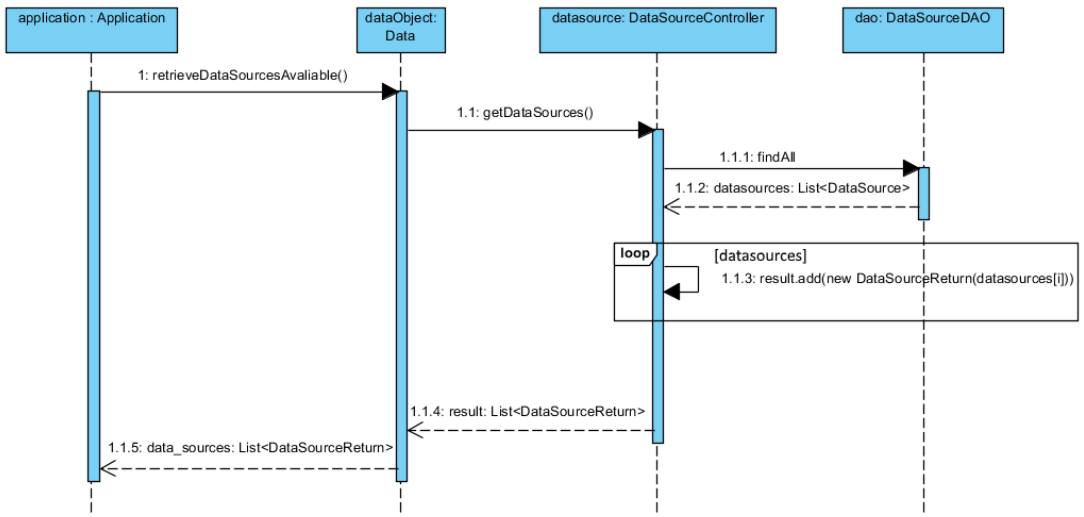


Ilustración 33 Obtención bases de datos disponibles

A continuación se repite el proceso pero utilizando el controlador de análisis de ADN, que a su vez empleará el dao instancia de DnaStudyDAO, encargado de obtener los análisis de ADN presentes en la base de datos.

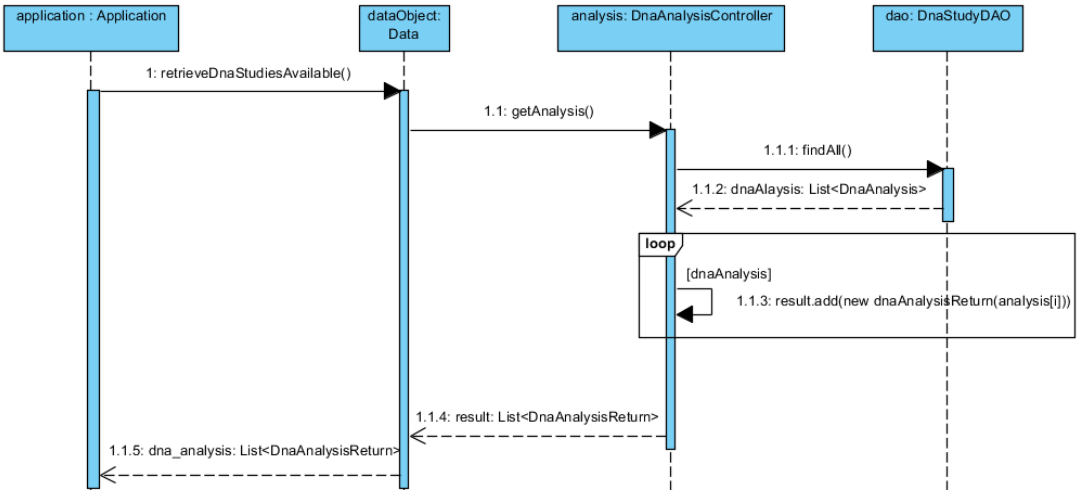


Ilustración 34 Obtención estudios ADN disponibles

Finalmente, el objeto aplicación posee los datos necesarios. Estos son enviados al objeto de tipo Session, que instancia un objeto de tipo FirstScreenDataSend poblándolo con los datos recibidos desde aplication. Se almacena la sesión y se envía el mensaje con los datos obtenidos; para ello, se utiliza un objeto de tipo mapper perteneciente a la librería Jackson que convierte el objeto JAVA en JSON para el correcto envío y recepción de este.

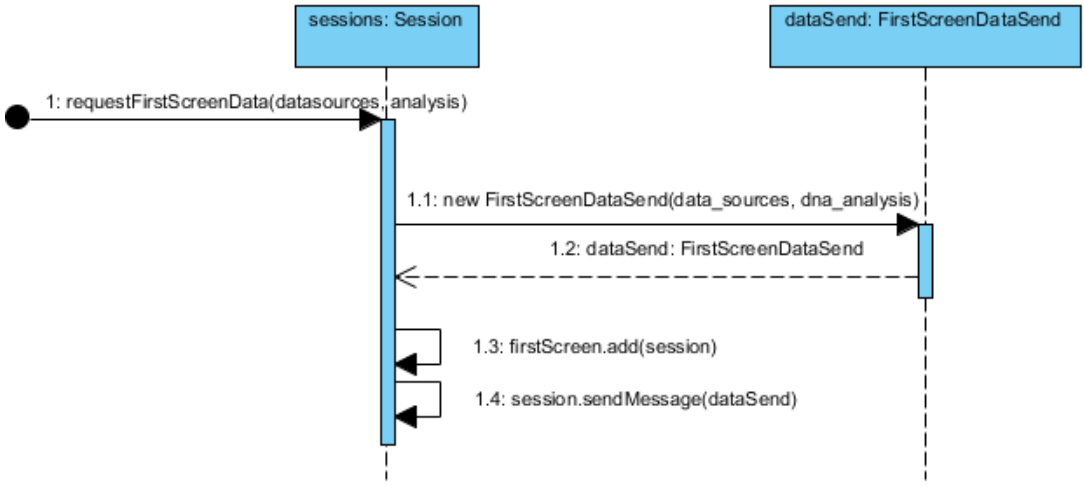


Ilustración 35 Envío de datos a la primera pantalla

Llegados a este punto, el usuario ya tiene visible tanto los estudios de ADN como las bases de datos disponibles; puede seleccionar los datos que considere oportunos y comunicárselo al servidor. Para la obtención de las variaciones con las que se va a trabajar entra en uso el objeto Data que, como se explicará en la sección 6.2, se encarga de gestionar los datos con los que va a trabajar el usuario. Cuando el servidor recibe este mensaje, lo deserializa para obtener los identificadores de las muestras pertenecientes a los estudios de ADN seleccionados y los identificadores de las bases de datos que van a ser utilizadas. Con esta información, el Objeto Data obtiene las variaciones de la vista creada en la base de datos mediante el controlador correspondiente. Una vez se han obtenido o actualizado (ya que los criterios de selección de variaciones pueden modificarse en cualquier momento) los datos el servidor se encarga de notificar a todos los widgets conectados los cambios que se han realizado en el conjunto de datos a mostrar.

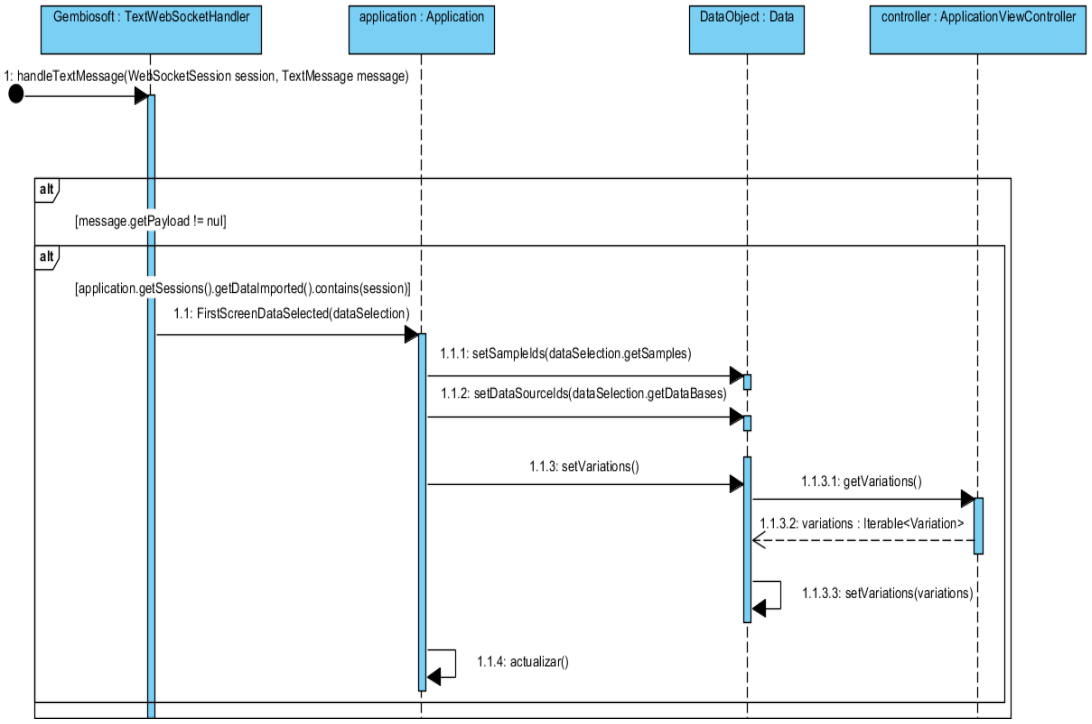


Ilustración 36 Obtención de variaciones

6.2. Gestión de los datos

Las variaciones cargadas se almacenan en el objeto Data, encargado de gestionarlas y modificar el conjunto si los criterios de selección son cambiados por parte del usuario. Por otra parte, la clase encargada de dar formato a los datos almacenados en Data es la clase Application. Por ejemplo, ante una petición de un widget de tipo chromosomeVsVariant., la clase Application agrupa las variaciones según su cromosoma y devuelve una lista de todos los cromosomas presentes con el número de variaciones que cumplen todos los criterios almacenados. Este dato es enviado al objeto Sessions que, además de almacenar las sesiones, es el encargado de comunicarse mediante mensajes con los widgets conectados al servidor a través de las sesiones.

El método más importante de los presentes en la clase Application es el método Actualizar. Este método es llamado siempre que se realiza una modificación en los criterios de selección de las variaciones, ya sea una modificación en la pantalla de selección de datos o la adición/eliminación de un filtro al conjunto de filtros presentes en la clase Filter. Cuando se llama este método, se obtienen las variaciones almacenadas en el objeto Data, vuelven a ser agrupadas según los criterios definidos y se dicha información es enviada a todos los widgets (excepto los de la pantalla de selección de

datos ya que estos no se modifican), consiguiéndose así una interacción en tiempo real de todos los usuarios conectados a la aplicación ante cualquier cambio.

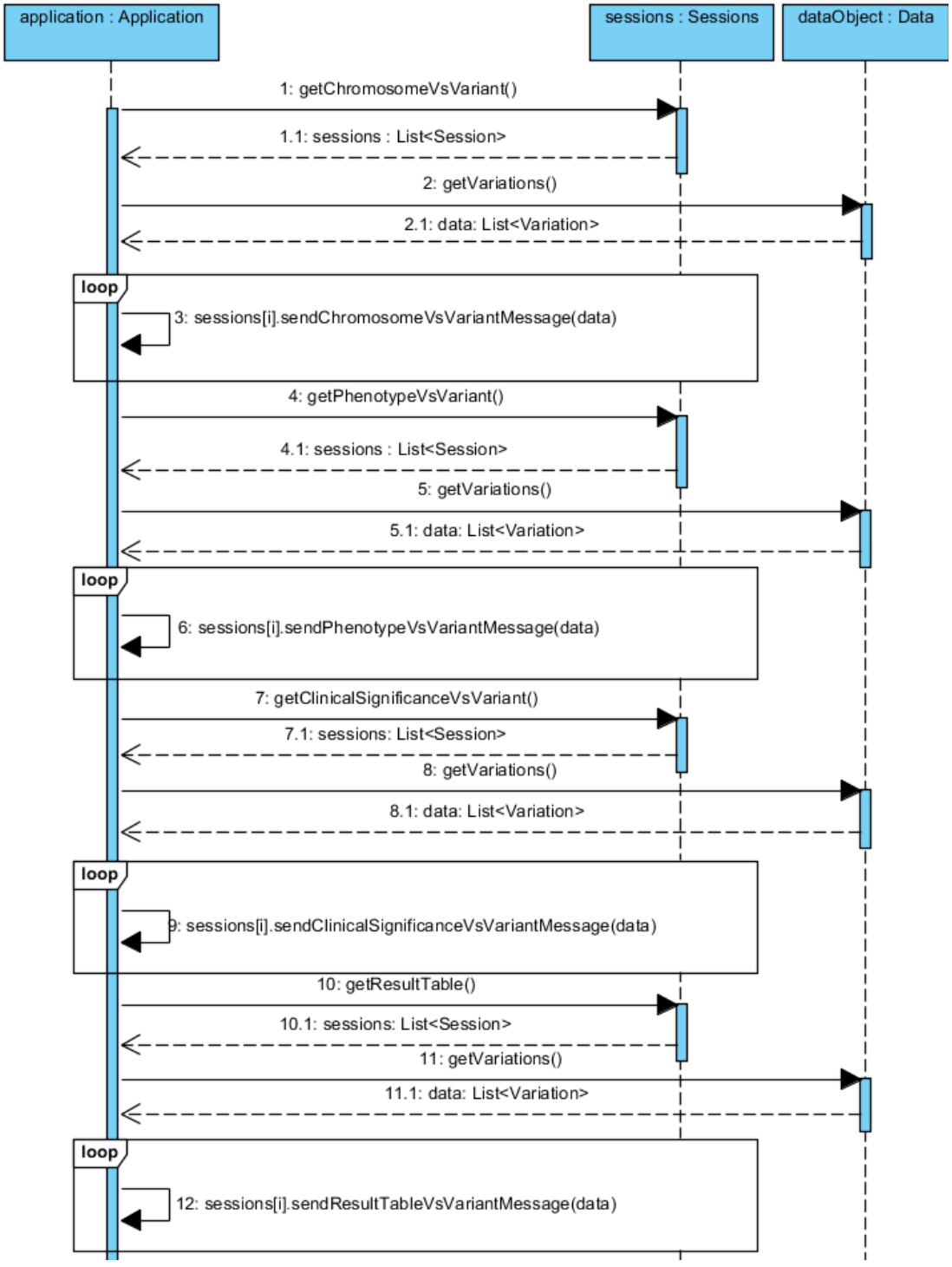


Ilustración 37 método actualizar. Etiqueta "actualizar"

6.3. Gestión de sesiones

Cuando un cliente se conecta al servidor bien para iniciar un análisis, bien para unirse a uno en curso; lo hace mediante un mensaje de los disponibles del API de la aplicación (ver sección 6.6). Este mensaje se envía a través de una sesión (ver sección 6.5). Si el mensaje es correcto y se trata de un mensaje que indique que es una nueva sesión, dicha sesión es almacenada en el sistema para recibir actualizaciones. Las sesiones se almacenan en un objeto de tipo Sessions. Este objeto está contenido en el objeto de tipo Application y se encarga no solo de gestionar las sesiones, sino de enviar los mensajes a las sesiones correspondientes cuando se realice una actualización. Cuando una sesión se cierra, el sistema se encarga de eliminar dicha sesión del conjunto de sesiones guardadas actualizando el número total de sesiones almacenadas.

Las sesiones pueden almacenarse en 7 listas distintas dependiendo del punto de origen de dicha conexión. Las sesiones se almacenan en función de la pantalla a la que pertenecen y, si pertenecen a la segunda pantalla, lo hacen en función del widget que están conectando al servidor. De este modo, Las diferentes listas en las que se puede almacenar una sesión son:

- dataImported: En esta lista se almacenan las sesiones creadas cuando la primera pantalla establece conexión con el servidor.
- sampleVsVariant: Cuando un widget de tipo sampleVsVariant se conecta al servidor, la sesión utilizada para dicha conexión se almacena en esta lista.
- chromosomeVsVariant: Cuando un widget de tipo sampleVsVariant se conecta al servidor, la sesión utilizada para dicha conexión se almacena en esta lista.
- phenotypeVsVariant: Cuando un widget de tipo sampleVsVariant se conecta al servidor, la sesión utilizada para dicha conexión se almacena en esta lista.
- clinicalSignificanceVsVariant: Cuando un widget de tipo sampleVsVariant se conecta al servidor, la sesión utilizada para dicha conexión se almacena en esta lista.
- reset: Cuando un widget de tipo sampleVsVariant se conecta al servidor, la sesión utilizada para dicha conexión se almacena en esta lista.
- resultTable*:* En esta lista se almacenan las sesiones creadas cuando la tercera pantalla establece conexión con el servidor.

6.4. Gestión de filtros

El usuario puede, mediante la selección de filtros, modificar el conjunto de datos visibles. El almacenamiento de estos criterios de filtrado se realiza en un objeto de tipo Filters que, como sucede con el objeto Sessions visto anteriormente, está contenido en el objeto Application. Estos filtros, dependiendo de su proveniencia, se almacenan en uno de las cuatro colecciones de filtros creadas:

- sampleIds: Este filtro corresponde al widget de tipo sampleVsVariant.
- chromosomeIds: Este filtro corresponde al widget de tipo chromosomeVsVariant.
- phenotypeIds: Este filtro corresponde al widget de tipo phenotypeVsVariant.
- clinicalSignificanceIds: Este filtro corresponde al widget de tipo clinicalSignificanceVsVariant.

Los mensajes gestionados por este objeto pueden ser de dos tipos. Por una parte, cuando se recibe un mensaje en el cual se añade un filtro nuevo; en cuyo caso el mensaje posee el formato de la clase FilterChange, que indica el identificador del elemento que se debe añadir al conjunto de filtros y el tipo al que pertenece dicho identificador.

```
{
  id="chr01",
  "type"="chromosome"
}
```

Ilustración 38 Ejemplo de mensaje con formato FilterChange

El sistema añade, en función del valor del atributo Type del mensaje recibido, el valor del atributo Id a la colección correspondiente y el sistema notifica a todas las sesiones el nuevo conjunto de datos que debe mostrar una vez se ha añadido el nuevo criterio al conjunto de filtros que debe superar una variación para poder ser visualizada.

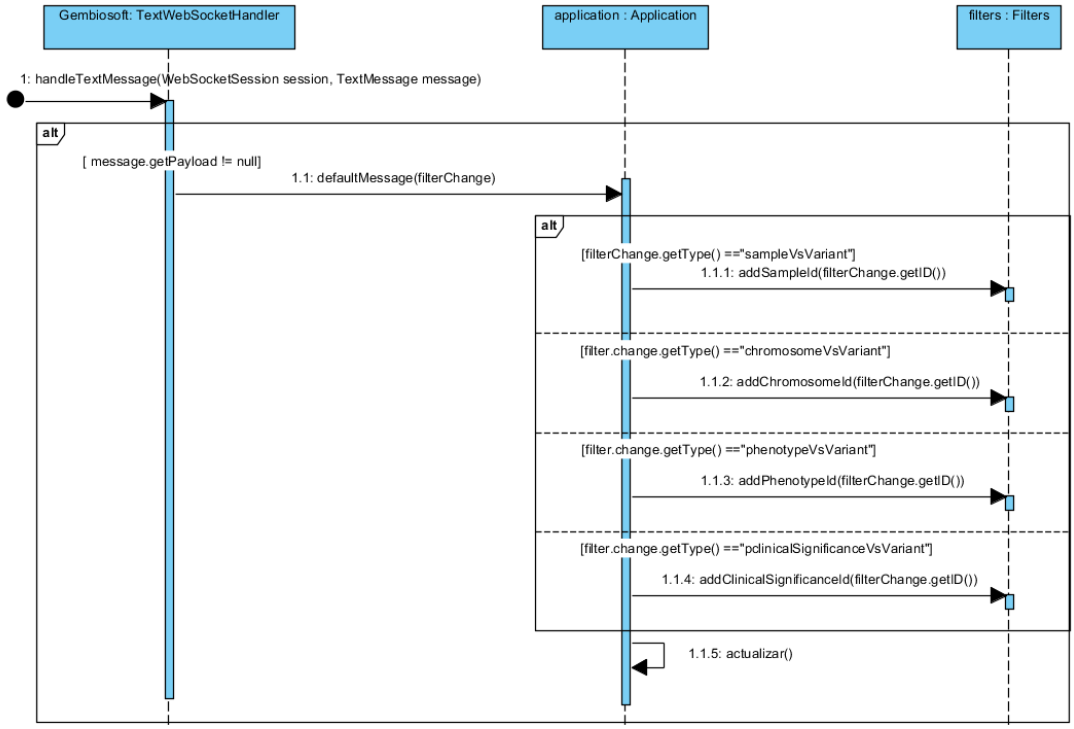


Ilustración 39 Ejemplo de mensaje añadiendo un filtro

Por otra parte, cuando se recibe un mensaje en el cual se elimina un filtro existente. Cuando se reciba un mensaje de una conexión que comunique el sistema con un widget de tipo reset, se mapea el objeto recibido (que sigue el mismo formato que el mensaje anterior, es decir, de la clase FilterChange). El valor del atributo Type identifica el tipo de filtro que debe ser eliminado y el valor del atributo Id identifica qué filtro debe ser eliminado. El valor del atributo Type puede ser sample, chromosome, phenotype o clinicalSigificance (Si hay que eliminar un filtro concreto perteneciente a la colección sampleId, chromosomeId, phenotypeId o clinicalSignificanceId respectivamente) o all, en cuyo caso se eliminan todos los filtros presentes.

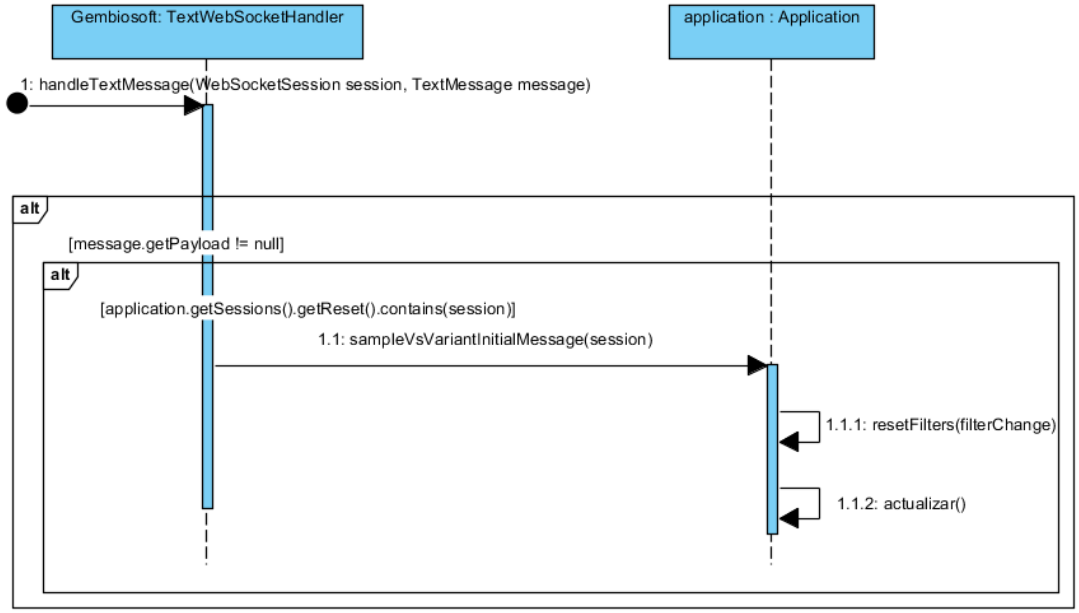


Ilustración 40 Ejemplo de mensaje para eliminar un filtro

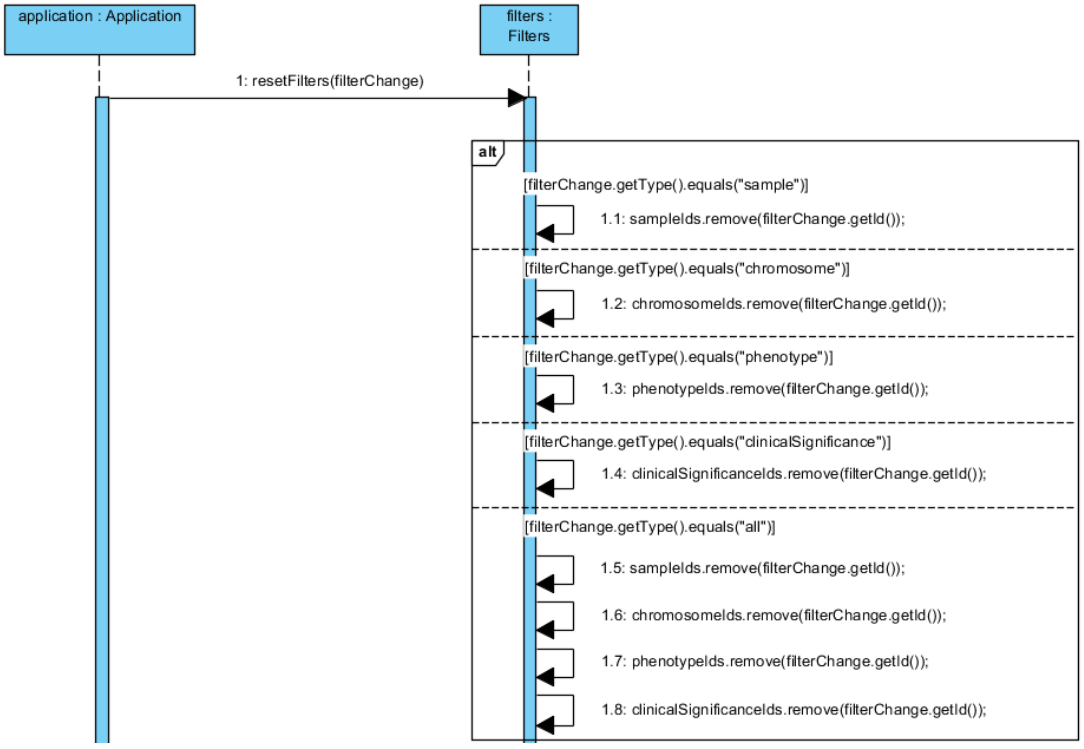


Ilustración 41 Ejemplo de eliminación de un filtro. Etiqueta "resetFilters".

Cada vez que se añade o se elimina un filtro, se notifica, mediante las sesiones existentes, el nuevo conjunto de datos que deben ser mostrados en la interfaz de usuario

mediante el envío de los datos formateados como correspondan (Véase método actualizar).

6.5. WebSockets

El *framework* de Spring proporciona su propia implementación del protocolo de comunicación WebSocket. Esta implementación está presente en el módulo spring-Websocket; esta implementación es compatible con el estándar JSR-356[40]. Crear un servidor que utilice Websockets es tan sencillo como implementar la interfaz WebSocketHandler. Esta interfaz es, a su vez implementada por las clases TextWebSockethandler y BinaryWebSocketHandler para mensajes de texto o binarios. Estas clases también pueden ser extendidas para crear Websockets. Para este trabajo se ha creado una clase llamada GemBiosoft que ha extendido a la clase TextWebSockethandler, modificando los siguientes métodos:

- afterConnectionEstablished: Cuando se establece conexión con el servidor desde el cliente este método se ejecuta. Debido a la naturaleza de la aplicación y a su implementación este método posee un cuerpo vacío. Recibe como parámetro la sesión establecida.
- handleTextMessage: Método ejecutado cuando se recibe un mensaje por parte del cliente. En el cuerpo de este método yace toda la lógica de negocio. Tras comprobar que el cuerpo del mensaje no está vacío, ejecuta el método o métodos necesarios según el contenido del mensaje. Recibe como parámetro el mensaje enviado para que pueda ser tratado por el servidor y la sesión por donde ha sido enviado dicho mensaje. Esta sesión puede ser utilizada bien para ser almacenada en las colecciones internas de la aplicación, bien para enviar un mensaje de respuesta por parte del servidor.
- afterConnectionClosed: Tras haber sido cerrada una sesión, se ejecuta este método, que se encarga de eliminar de las colecciones de sesiones la referencia a dicha sesión para evitar errores de envío en el futuro que puedan alterar el normal funcionamiento de la aplicación. Recibe como parámetro la sesión que ha sido cerrada y el motivo por el cual ha sido cerrada.
- handleTransportError: Cuando se produce un error en el transporte de un mensaje a través de una sesión, se ejecuta el método, el cual indica la sesión

[40] Se trata de la API estándar de JAVA para Websockets.

en la que se ha producido el fallo y su motivo. Recibe como parámetro la sesión donde ocurrió el error y la excepción sufrida durante el envío del mensaje.

Una vez se ha implementado la clase referente a la lógica del Websocket, se debe realizar la configuración necesaria para que el Websocket funcione correctamente y pueda ser accedido desde el exterior. Para ello, se ha implementado la clase de Spring WebSocketConfigurer; esta interfaz define los métodos para configurar las peticiones a los Websockets del servidor. Gracias al uso de las anotaciones el proceso es simple y rápido, a dicha clase se le añaden las anotaciones @Configuration para indicar que se declaran métodos que utilizan Beans y su comportamiento en tiempo de ejecución; y @EnableWebSocket para indicar que se está configurando el proceso de peticiones a websockets. Posteriormente se define un vean mediante la anotación @Bean de la clase implementada antiormente. En este caso, un Bean de tipo WebSocketHandler de la clase GemBiosoft.

Finalmente, se modifica el método registerWebSocketHandler que registra un WebSocketHandler [41] . Recibe como parámetro un atributo de tipo WebSocketHandleRegistry. Esta clase provee métodos para los mapeos de solicitudes a websockets. En el cuerpo de este método se ha añadido al WebSocketHandleRegistry un nuevo handler donde se mapea la dirección con el Bean creado.

[41] Se trata de un *handler* para gestionar los mensajes y los eventos del ciclo de vida de un WebSocket.

```
@Configuration
@EnableWebSocket
public class MyWebSocketConfig implements WebSocketConfigurer {

    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {

        registry.
        addHandler(Gembiosoft(), "/gembiosoft").
        addInterceptors(new HttpSessionHandshakeInterceptor()).
        setAllowedOrigins("*");
    }

    @Bean
    public WebSocketHandler Gembiosoft() {
        return new Gembiosoft();
    }
}
```

Ilustración 42 Configuración WebSocket

6.6. API

La comunicación entre la interfaz de usuario y la lógica de negocio se realiza mediante el intercambio de mensajes en formato JSON a través de las conexiones establecidas mediante Websockets. Tanto los mensajes de peticiones como los de respuesta poseen un formato claramente definido creando así una API para la recuperación de los datos. Para utilizar la librería Jackson se ha añadido la dependencia necesaria al archivo pom.xml. Esta, consta de grupo "com.fasterxml.jackson.core" y artefacto "Jackson core".

Cuando se realiza una petición al servidor o se realiza un cambio que afecta al resto de componentes se envía un mensaje de texto a través de la conexión establecida entre el widget y el servidor. Empezaremos nombrando las peticiones que se pueden realizar al servicio así como su respuesta y a continuación detallar las respuestas a dichas peticiones para terminar con el formato seguido cuando es el cliente quien comunica que se ha realizado un cambio.

Así pues, pasamos a describir dicha API estructurando las peticiones según la pantalla en que se realicen. En primer lugar, para la primera pantalla (selección de los datos con los que vamos a trabajar), disponemos de las siguientes peticiones:

Un mensaje cuyo contenido sea “firstScreen” permite al servidor identificar dicha conexión almacenándola en la colección que guarda los identificadores de la primera pantalla. Este mensaje debe ser enviado cuando se establece conexión por primera vez entre el servidor y la primera pantalla para recibir así los datos.


```
{
  "dataList": [
    { "id": "dbSNP", "name": "The SNP database" }
  ],
  "dnaList": [
    {
      "name": "NSD1 Father-son Study (F2)",
      "description": "A predictive analysis to check predisponsition of a son to heart failu
      "samples": [
        { "id": 96, "name": "Father of family F2", "size": 7 }
      ],
      "id": 88
    }
  ]
}
```


Ilustración 43 Ejemplo respuesta a petición "firstScreen"

El mensaje JSON creado por el servidor como respuesta contiene dos listas; la primera contiene el identificador y nombre de todas las bases de datos disponibles, la segunda el nombre, descripción, muestras y número de muestras de todos los estudios existentes en la base de datos. Cabe destacar que cada muestra incluye su identificador, nombre y el número de variaciones que contiene. Una vez el usuario ha seleccionado los datos que desea emplear para en el diagnóstico, se envía un nuevo mensaje en formato JSON siguiendo el formato de representación de la clase interna DataSelection con la lista de identificadores seleccionados. El servidor identifica que el mensaje pertenece a la primera pantalla y procede a obtener de la base de datos las variaciones que pertenezcan a la selección realizada.


```
{"samples":[96], "databases":["dbSNP"]}
```


Ilustración 44 Ejemplo selección del usuario

En segundo lugar abordamos tanto las peticiones como el intercambio de mensajes de la segunda pantalla. Cuando un widget establece conexión con el servidor envía un mensaje identificándose para saber qué información enviar cuando los datos a mostrar

deban ser actualizados por un cambio en el conjunto de filtros aplicado a los datos. Se ha definido un mensaje por cada tipo de widget:

Para el caso de los gráficos, por cada tipo de gráfico disponible (variante por cromosoma, variante por fenotipo y variante por significancia clínica) se dispone de un mensaje que permita identificar qué información contiene ("chromosomeVsVariant", "phenotypeVsVariant" y "clinicalSignificanceVsVariant" respectivamente) y el formato que deben seguir los datos que se envíen. Cuando el servidor recibe uno de estos mensajes almacena la conexión en la colección correspondiente y envía los datos para que sean representados gráficamente. En los tres casos se devuelve un mapa con las claves del elemento de interés (cromosoma, fenotipo o significancia clínica) y su valor (número entero que determina el número total de variaciones que pertenecen a dicha clave y que además cumplen con todas las posibles restricciones aplicadas con los filtros).

```
{"chr18":1,"chr5":16,"chr17":10,"chr1":17,"chr13":48}
```

Ilustración 45 Ejemplo respuesta "chromosomeVsVariant"

Puesto que inicialmente todavía no se ha aplicado ningún filtro, se cargan todas las variaciones del conjunto de datos cargados a partir de los estudios y bases de datos seleccionados en la primera pantalla.

Una vez han sido poblados, es posible hacer clic en uno de los elementos del gráfico para filtrar por dicho elemento. Cuando se realiza esta acción se envía un mensaje siguiendo la estructura de la clase FilterChange, que consta de dos campos, uno con el identificador el elemento seleccionado y otro con el tipo de identificador seleccionado.

Cuando llega uno de estos mensajes al servidor, este se encarga de añadir el filtro y actualiza la información a mostrar en todos los widgets. Cuando se realiza la actualización el servidor recorre las colecciones de conexiones guardadas y para los elementos de cada colección reenvía la información actualizada según el conjunto de filtros existente.

```
{"id":"chr5", "type":"chromosomeVsVariant"}
```

Ilustración 46 Ejemplo mensaje para filtrar las variaciones por cromosoma

Debido a la gran cantidad de muestras que se pueden manejar a lo largo del diagnóstico y a su distinta naturaleza en comparación con el resto de widgets

(normalmente van a haber varias muestras seleccionadas todo el tiempo reduciendo la usabilidad de la tabla de los filtros), estas se representan en una tabla a parte donde cada elemento de la tabla posee un checkbox por lo tanto su funcionamiento varía respecto al resto de widgets. Para obtener los datos necesarios para poblar esta tabla es necesario enviar un mensaje cuyo cuerpo sea "sampleVsVariant". Cuando el servidor recibe este mensaje almacena la sesión y envía dos mensajes: Por un lado envía el conjunto de muestras con su nombre y la cantidad de variaciones que posee para poblar la tabla. Por otro lado, envía el conjunto de muestras seleccionadas en dicho momento (en el caso de que devuelva el conjunto vacío se asume que están todas seleccionadas) para así saber que *checkbox* debe marcar.

{"MTH_FM1":17,"DGT_FM1":39,"FTH_FM2":10,"FTH_FM1":14,"SN_FM2":6,"SN_FM1":6}

Ilustración 47 Ejemplo respuesta a mensaje "sampleVsVariant"

Los filtros seleccionados se ven reflejados en una tabla. En el momento de su creación se establece una conexión vía Websocket y envía un mensaje cuyo contenido es "reset". El servidor almacena la conexión en la colección adecuada; a continuación, envía como respuesta el conjunto de filtros que están siendo utilizados (excluyendo las muestras).


```
{
  "Filters": [
    "chromosomeIds": [
      "chr5"
    ],
    "phenotypeIds": [

    ],
    "clinicalSignificanceIds": [

    ]
  ]
}
```


Ilustración 48 Ejemplo respuesta a mensaje "reset"

Cuando se opta por eliminar uno de los filtros existentes se envía un mensaje al servidor siguiendo la estructura de la ya mencionada antes clase FilterChange; según lo seleccionado en la interfaz gráfica de usuario el mensaje puede estar eliminando una

muestra, en cuyo caso el atributo type tendrá el valor simple y su atributo valor será la id de la muestra; un cromosoma, fenotipo o significancia clínica (con el atributo type con valor chromosome, phenotype o clinicalSignificance respectivamente) o todos los filtros existentes si el valor del atributo type es all. En este último caso el campo id está vacío.

Finalmente, la tercera pantalla posee un único tipo de petición: "resultTable" para identificarse como la tabla de la última pantalla. El servidor, al recibir este mensaje, almacena la conexión y envía una lista con todas las variaciones que cumplen los filtros establecidos para ser presentados al usuario.


```
[
  {
    "dna_study_id": 87,
    "dna_study_name": "BRCA Family Study (F1)",
    "variation_id": 110,
    "chromosome_id": 28,
    "chromosome_name": "chr5",
    "chromosome_position": 31829692,
    "reference_allele": "A",
    "alternative_allele_a": "G",
    "alternative_allele_b": null,
    "sample_id_db": 94,
    "sample_id": "DGT_FM1",
    "clinical_significance_id": 7,
    "clinical_significance": "not provided",
    "phenotype_id": null,
    "phenotype_name": null,
    "phenotype_description": null,
    "variation_id_in_data_source": "rs9292444\r\n",
    "data_source_id": "dbSNP",
    "data_source_name": "The SNP database",
    "gene_id": 101,
    "gene_name": "PDZD2",
    "research_publication_id_db": null,
    "publication_site": null,
    "research_publication_id": null,
    "research_publication_link": null
  }
]
```


Ilustración 49 Ejemplo respuesta a mensaje "resultTable"

En resumen, el servicio de recuperación de datos que se encarga de la recuperación de los datos consta de 7 peticiones: "firstScreen", "reset", "sampleVsVariant", "chromosomeVsVariant", "phenotypeVsVariant", "clinicalSignificanceVsVariant" y

"resultTable" y puede recibir información en dos formatos distintos, mapeados por las siguientes clases: FilterChange y DataSelection.

Si el mensaje enviado desde el cliente es vacío se responde advirtiendo de que el mensaje es nulo y no se realizará acción alguna; del mismo modo, ante un mensaje formateado erróneamente no se realizará acción algún excepto advertir al usuario. De este modo se cubren posibles errores por parte del cliente a la hora invocar la API de la aplicación o interaccionar con esta.

6.7. Acceso seguro al servidor

La seguridad es un elemento de vital importancia cuando se trata de aplicaciones *cloud* que trabajan con grandes datos algunos de los cuales son confidenciales. Es aconsejable asegurar una conexión segura entre el servidor y los distintos clientes. Se ha utilizado un certificado autofirmado generado con el software OpenSSL y, una vez configurado, almacenarlo en un almacén de claves creado anteriormente mediante la herramienta JAVA keytool.

Spring, y más concretamente Spring boot, facilita en gran medida la configuración a realizar para que el servidor utilice conexión cifrada mediante SSL. Para ello, tan solo hay que añadir una serie de líneas al archivo application.properties autogenerado cuando se crea el proyecto. Las propiedades importadas de server.sll.* proporcionan múltiples opciones de configuración, sin embargo, con las opciones básicas será suficiente. Las propiedades configuradas son las siguientes:

- server.port = 8443
  Identifica el puerto que se va a utilizar para acceder a los servicios. El puerto 443 es el puerto utilizado por omisión en el protocolo HTTPS
- server.ssl.key-store = classpath:keystore.jks
  Ruta que almacena el certificado autofirmado generado mediante OpenSSL
- server.ssl.key-store-password = *****
  Contraseña utilizada para almacenar nuevas claves en el depósito de claves.
- server.ssl.key-password = ****
  Contraseña utilizada para acceder al depósito de claves

Cabe destacar que si se desea soporte tanto para HTTP como para HTTPS será necesario configurar uno de ellos de manera programática ya que el archivo application.properties no permite la configuración declarativa de más de un conector.

Se puede observar como gracias a Spring Boot se puede permitir el acceso seguro mediante SSL a un servidor de una manera rápida y elegante añadiendo tan solo unas pocas propiedades al proyecto.

6.8. Configuración y acceso del servidor

Para acceder al servidor basta con seguir la sintaxis propia de las conexiones de WebSockets. Cabe mencionar que se ha utilizado un servidor Apache Tomcat para desplegar el servicio y se han utilizado los puertos por defecto. Esta, según sea mediante protocolo HTTP o HTTPS, sigue la siguiente estructura:

- Ante protocolo HTTP: ws:dirección_ip:puerto/nombre_del_servicio. De este modo, la dirección es: ws:158.42.185.198:8080/gembiosoft
- Ante protocolo HTTPS: wss:dirección_ip:puerto/nombre_del_servicio. De este modo, la dirección es: wss:158.42.185.198:8443/gembiosoft

Es posible cambiar el puerto por defecto desde el archivo application.properties para asignarle cualquier otro puerto disponible.

6.9. Despliegue

Una vez desarrollados los servicios, es hora de desplegar la aplicación. Este despliegue ha sido realizado en dos pasos; el primero ha consistido en crear y configurar una instancia en Fiware para alojar la base de datos y el servidor Wirecloud. Por otra parte, el segundo paso ha sido desplegar los servicios de recuperación de datos en un servidor Apache externo a Fiware.

Empezando por Fiware, el primer paso ha consistido en obtener una IP pública para asignarla a la instancia y una *keypair*[42] para poder acceder desde el exterior a la instancia. A continuación, se ha creado un *blueprint* al cual se ha añadido del catálogo disponible JAVA 1.7 y Apache Tomcat 6 (una de las ventajas de crear un *blueprint*, además de permitir añadir software del catálogo para que se instale de manera transparente, es la creación automática de un grupo de seguridad asignado a la instancia con las reglas de seguridad ya definidas). Una vez desplegada la instancia y con el software del catálogo

[42] Conjunto de claves criptográficas pública/privada utilizadas para encriptar el acceso externo a una sesión

instalado correctamente, han sido instalados dos componentes: Por un lado se ha instalado PostgreSQL y se ha migrado la base de datos a la instancia. Por otro lado, para instalar el servidor Wirecloud, fue necesario instalar manualmente Pyhton (Lenguaje de programación multiparadigma) y Django (*framework* de desarrollo web escrito en python). Las interfaces de usuario (las tres panatallas que presenta la aplicación) fueron cargadas como un sitio web en Apache Tomcat. La primera y tercera interfaz (carga de datos y presentación de resultado respectivamente) contienen webcomponents realizados en Polymer (Biblioteca para el desarrollo *front-end* de componentes web que combina HTML, CSS y JAVAScript). La segunda interfaz (tratamiento y filtrado de datos) contiene un frame que llama al portal de Wirecloud; esto quiere decir que una parte de la interfaz visualizada provenía del servidor de Wirecloud instalado en Fiware.

Finalmente, solo resta el despliegue del servicio de recuperación de datos. Para ello, ha bastado con exportar el servidor y colocarlo en un servidor Apache Tomcat. Se ha empleado Maven para empaquetar el contenido del proyecto Spring. La lista de dependencias utilizadas nos muestra la dependencia que ha hecho posible este empaquetamiento mediante Maven: Spring-boot-maven-plugin proporciona soporte para Maven, haciendo posible empaquetar un proyecto de Spring boot en un archivo ejecutable war o jar. Cabe destacar que Spring-boot-starter-tomcat se debe indicar como provided para no obtener ningún error. Tras preparar el proyecto para que pueda ser empaquetado, se ejecuta la orden correspondiente de maven; para ello, ejecutamos la aplicación como Maven build indicando como goal (objetivo) "clean install" para compilar y empaquetar el código del proyecto, dejando el archivo war en el repositorio local. Una vez obtenido el archivo war, solo hay que moverlo al directorio correspondiente de Apache Tomcat.

7. Conclusiones

A lo largo de este trabajo de fin de grado se ha realizado el desarrollo de un servicio de recuperación y gestión de datos como parte de una aplicación de web colaborativa para la elaboración de diagnóstico médico y su posterior despliegue en la plataforma *cloud* Fiware realizando un meticuloso análisis de las capacidades ofrecidas por esta plataforma.

El primero de los objetivos consistió en el desarrollo de los servicios descritos. Como consecuencia de lo expuesto a lo largo de la memoria se concluye un exitoso desarrollo de estos servicios cumpliendo con los objetivos iniciales:

Estos servicios ofrecen un entorno concurrente y colaborativo que permite interactuar simultáneamente a varios usuarios a través de la interacción con la aplicación. Ha sido creada una arquitectura que, a pesar del intenso flujo de intercambio de mensajes entre el servidor y los múltiples clientes, ofrece una interacción en tiempo real capaz de actualizar de manera instantánea todos los clientes ante cualquier cambio registrado en el servidor. Aunque debido a la naturaleza de los Websockets, estos presentan una leve limitación en cuanto a la calidad de la conexión, ya que, si no ofrece conectividad perfecta, estos no serán funcionales.

Gracias al uso de Fiware y a la naturaleza de las tecnologías empleadas (tanto PostgreSQL como los Websockets han sido desarrolladas con el objetivo de ofrecer una gran flexibilidad y concurrencia) se ha dado lugar a una aplicación con una alta tolerancia al crecimiento exponencial de los datos disponibles (uno de los grandes problemas en el ámbito de la bioinformática) y escalable mediante el uso de Fiware como plataforma *cloud*. Se ha testeado la aplicación web con un total de 10 dispositivos, incluyendo ordenadores de sobremesa, portátiles y televisiones, conectados simultáneamente y no se ha observado ningún tipo de retraso o demora manteniéndose fluido en todos los dispositivos.

Otro de los puntos fuertes de la aplicación es su gran simplicidad; pensando en el usuario final, que no tiene por qué tener conocimientos informáticos ni los necesita para utilizarla. Se proporciona una interfaz limpia y definida con un conjunto de funcionalidades intuitivas y una retroalimentación continua. Diversas pruebas de usabilidad fueron realizadas con voluntarios que poseían un conocimiento informático bajo y en el 100% de los casos se

obtuvo una experiencia de uso positiva tanto en la funcionalidad de la aplicación como la claridad de los resultados.

El segundo objetivo refiere al análisis de la *cloud* Fiware. La experiencia global de Fiware, si bien no ha sido negativa, ha presentado suficientes inconvenientes como para, en el futuro, intentar evitar el uso de esta plataforma y optar por otras opciones más asentadas en el mercado y con más experiencia. Si bien sobre el papel es un proyecto interesante, en este momento es un proyecto al que todavía le falta mucho camino por recorrer (especialmente en un campo donde la competencia existente es tan fuerte, con competidores como Amazon Web Services por ejemplo) si realmente pretende convertirse en referencia a nivel mundial.

Si bien este TFG ha concluido, no es el caso del proyecto, pues sigue su desarrollo con una clara visión de futuro. Se plantea un emocionante horizonte donde no solo se va a seguir desarrollando una aplicación que, por su concepto y las tecnologías empleadas, es realmente emocionante, sino que además ofrece la posibilidad de intentar ayudar a los profesionales de un campo tan importante como es la medicina, poniendo nuestro pequeño grano de arena en la lucha contra el cáncer y otras enfermedades relacionadas de una forma u otra con la genética. Por este motivo ya se está trabajando en el siguiente paso, consistiendo en el desarrollo de herramientas ETL[43] para empezar a poblar la base de datos con variaciones reales obtenidas de fuentes de datos abiertas como por ejemplo ClinVar.

[43] Una herramienta ETL (Extract, Transform, Load) permite migrar datos desde múltiples fuentes, reformatearlos y cargarlos en otras fuentes de datos

8. Bibliografía

[1] Y. Benkler y H. Nissenbaum, «Commons-based peer production and virtue*», J. Polit. Philos., vol. 14, n.º 4, pp. 394–419, 2006.

[2] Y. Benkler, The wealth of networks: how social production transforms markets and freedom. New Haven [Conn.]: Yale University Press, 2006.

[3] Roberto Acuña, María Antonieta Campos, y Mayela Dabdub, «aprendiendodemaneracolaborativa - Importancia del trabajo colaborativo para el pensamiento crítico», jul-2010. [En línea]. Disponible en: https://aprendiendodemaneracolaborativa.wikispaces.com/Importancia+del+trabajo+colaborativo+para+el+pensamiento+cr%C3%ADtico.

[4] «Collaborative workspace», Wikipedia, the free encyclopedia. 07-ene-2016.

[5] Peter and Trudy Johnson-Lenz, «Rhythms, Boundaries, and Containers: Creative Dynamics of Asynchronous Group Life», pp. 395-417, abr. 1990.

[6] «ABOUT US » FIWARE». [En línea]. Disponible en: https://www.fiware.org/about-us/.

[7] «Welcome to the FIWARE Wiki - FIWARE Forge Wiki». [En línea]. Disponible en: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Main_Page.

[8] «PostgreSQL», Wikipedia, la enciclopedia libre. 30-abr-2016.

[9] «PostgreSQL: About». [En línea]. Disponible en: http://www.postgresql.org/about/.

[10] «What's New in JDK 8». [En línea]. Disponible en: http://www.oracle.com/technetwork/java/java/8-whats-new-2157071.html.

[11] «JSON Tutorial». [En línea]. Disponible en: http://www.w3schools.com/json/.

[12] «FasterXML/jackson-core», GitHub. [En línea]. Disponible en: https://github.com/FasterXML/jackson-core.

[13] J. Dreyfuss, «The Ultimate JSON Library: JSON.simple vs GSON vs Jackson vs JSONP», Takipi Blog, 28-may-2015. [En línea]. Disponible en: http://blog.takipi.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/.

[14] «JacksonHome - FasterXML Wiki». [En línea]. Disponible en: http://wiki.fasterxml.com/JacksonHome.

[15] C. Álvarez, «JPA vs Hibernate», Genbeta Dev, 22-jul-2014. [En línea]. Disponible en: http://www.genbetadev.com/frameworks/jpa-vs-hibernate.

[16] «Java Persistence API». [En línea]. Disponible en: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html.

[17] «Hibernate. Everything data. - Hibernate». [En línea]. Disponible en: http://hibernate.org/.

[18] «JBoss Tools - Home». [En línea]. Disponible en: http://tools.jboss.org/.

[19] Vanessa Wang, Frank Salim, y Peter Moskovits, The definitive Guide to HTML5 WebSocket. 2013.

[20] I. Fette y A. Melnikov, «The websocket protocol», 2011.

[21] «Escribir servidores WebSocket», Mozilla Developer Network. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/WebSockets-840092-dup/Escribiendo_servidores_con_WebSocket..

[22] «HTTP/1.1: Status Code Definitions». [En línea]. Disponible en: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html.

[23] C. Schaefer, C. Ho, y R. Harrop, Pro Spring, 4th ed. Berkeley, CA: Apress, 2014.

[24] Scott Deeg, «Spring boot», 18:54:21 UTC.

[25] C. Álvarez, «¿Qué es Maven?», Genbeta Dev, 31-ago-2014. [En línea]. Disponible en: http://www.genbetadev.com/java-j2ee/que-es-maven.

[26] «Maven – Welcome to Apache Maven». [En línea]. Disponible en: https://maven.apache.org/..

[27] «Building Java Projects with Maven». [En línea]. Disponible en: https:// spring.io/guides/gs/maven/.

[28] «What is an SSL Certificate?» [En línea]. Disponible en: https://www.globalsign.com/en/ssl-information-center/what-is-an-ssl-certificate/.

[29] «¿Qué es SSL (Secure Sockets Layer)? - DigiCert». [En línea]. Disponible en: https://www.digicert.com/es/ssl.htm.

[30] A. Vukotic y J. Goodwill, Apache Tomcat 7. Berkeley, CA: Apress, 2011.

[31] M. Zwolak y M. Di Ventra, «Physical approaches to DNA sequencing and detection», Rev. Mod. Phys., vol. 80, n.º 1, pp. 141-165, ene. 2008.

[32] M. J. Villanueva Del Pozo, «An agile model-driven method for involving end-users in DSL development», 2016.

[33] M. J. V. D. Pozo, «Diagen: Modelado e Implementación de un framework para el análisis personalizado del ADN», jul. 2011.

[34] «FI-WARE - HackForGood». [En línea]. Disponible en: http://cattelefonica.webs.upv.es/Fiware/FIWARE.pdf.

[35] «FIWARE Frequently Asked Questions (FAQ) - FIWARE Forge Wiki». [En línea]. Disponible en: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Frequently_Asked_Questions_(FAQ).

[36] «fiware: Course categories». [En línea]. Disponible en: https://edu.fiware.org/course/index.php.

[37] «FIWARE Lab». [En línea]. Disponible en: http://infographic.lab.fiware.org/.

[38] J. Camenisch, «Concepts Around Privacy-Preserving Attribute-Based Credentials», en Privacy and Identity Management for Emerging Services and Technologies, M. Hansen, J.-H. Hoepman, R. Leenes, y D. Whitehouse, Eds. Springer Berlin Heidelberg, 2013.

[39] «FIWARE OPS Tools - FIWARE Forge Wiki». [En línea]. Disponible en: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_OPS_Tools.

[40] Silvio Cretti, «D.21.1.1: Platform deployment, operations, analytics and support tools. Future Internet - Core». 30-sep-2015.

[41] «fidash/fiware-fidash», GitHub. [En línea]. Disponible en: https://github.com/fidash/fiware-fidash.

[42] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, y M. Xu, «Web services agreement specification (WS-Agreement)», en Open Grid Forum, 2007, vol. 128, p. 216.

[43] «Atos-FiwareOps/sla-framework», GitHub. [En línea]. Disponible en: https://github.com/Atos-FiwareOps/sla-framework.

[44] «Atos-FiwareOps/sla-dashboard», GitHub. [En línea]. Disponible en: https://github.com/Atos-FiwareOps/sla-dashboard.

[45] «telefonicaid/fiware-health», GitHub. [En línea]. Disponible en: https://github.com/telefonicaid/fiware-health.

[46] «SmartInfrastructures/fi-lab-infographics», GitHub. [En línea]. Disponible en: https://github.com/SmartInfrastructures/fi-lab-infographics.

[47] «Atos-FiwareOps/flavor-sync», GitHub. [En línea]. Disponible en: https://github.com/Atos-FiwareOps/flavor-sync.

[48] «telefonicaid/fiware-glancesync», GitHub. [En línea]. Disponible en: https://github.com/telefonicaid/fiware-glancesync.

[49] «Atos-FiwareOps/fiware-maintenance-calendar-api», GitHub. [En línea]. Disponible en: https://github.com/Atos-FiwareOps/fiware-maintenance-calendar-api.

[50] «telefonicaid/fiware-skuld», GitHub. [En línea]. Disponible en: https://github.com/telefonicaid/fiware-skuld.

[51] «Reference Nodes Accelerators - FIWARE Forge Wiki». [En línea]. Disponible en: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Reference_Nodes_Accelerators.

[52] «Mundus » FIWARE». [En línea]. Disponible en: https://www.fiware.org/mundus/.

[53] «Mundus Region » FIWARE». [En línea]. Disponible en: https://www.fiware.org/mundus-regions/.

[54] «FIWARE Lab Nodes Handbook - FIWARE Forge Wiki». [En línea]. Disponible en: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Lab_Nodes_Handbook.

[55] E. Evans, Domain-driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional, 2004.