Facultad
de
Ciencias

INTERFAZ DE ANÁLISIS DE DATOS MEDIANTE GRÁFICAS

(Data analysis interface by charts)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Carlos Lanza Bolado

Director: Carlos Blanco Bueno

Co-Director: Daniel San Martín Segura

Marzo – 2016

Índice

1. Introducción 1
1.1. Introducción y contexto del proyecto 1
1.2. Motivación del trabajo a realizar 1
1.3. Objetivos 2
2. Material y métodos utilizados 3
2.1. Metodología 3
2.2. Planificación del trabajo 4
2.3. Tecnologías, lenguajes y herramientas 5
2.3.1. JAVA 5
2.3.2. Vaadin 5
2.3.3. HighCharts 6
2.3.4. MySQL 7
2.3.5. Librerías Predictia 7
2.3.6. Predicados Guava 7
2.3.7. Subversion 8
2.3.8. Tomcat 8
2.3.9. Eclipse 9
3. Presentación del problema y análisis de requisitos 10
3.1. Requisitos funcionales 10
3.2. Requisitos no funcionales 11
3.3. Diagrama de casos de uso 12
4. Diseño e implementación 13
4.1. Diseño arquitectónico 13
4.2. Diseño detallado e implementación 13
4.2.1. Capa de presentación 13
4.2.2. Capa de negocio 15
4.2.3. Capa de persistencia 30
5. Pruebas 30
5.1. Pruebas unitarias 30
5.2. Pruebas de integración 32
5.3. Pruebas de sistema 32
5.3.1. Pruebas de usabilidad 32
5.3.2. Pruebas de rendimiento 33
5.4. Pruebas de aceptación 33
6. Ejemplo Práctico 33

7. Conclusiones y trabajos futuros 42
7.1. Conclusiones 42
7.2. Trabajos futuros 42
7.2.1 Traducción de la interfaz 42
7.2.2. Exportación de las gráficas 43
7.2.3. Añadir más tipos de gráficas 43
8. Agradecimientos 43
Bibliografía y referencias 44

Índice de imágenes

Imagen 1: Planificación del trabajo 4
Imagen 2: Arquitectura cliente-servidor de Vaadin 6
Imagen 3: Diagrama de Casos de Uso 12
Imagen 4: Menú inicial 13
Imagen 5: Creación del menú de la interfaz 14
Imagen 6: Generación del menú de gráficas 14
Imagen 7: Estructura código Primer menú 15
Imagen 8: Estructura código menú gráfica 15
Imagen 9: Estructura Capa de negocio 15
Imagen 10: Diagrama de clases del paquete color 17
Imagen 11: Diagrama de clases del paquete utils 17
Imagen 12: Diagrama de clases del paquete properties - 1 18
Imagen 13: Diagrama de clases del paquete properties - 2 19
Imagen 14: Diagrama de clases del paquete properties - 3 20
Imagen 15: Diagrama de clases del paquete charts - 1 21
Imagen 16: Diagrama de clases del paquete charts - 2 22
Imagen 17: Diagrama de clases del paquete charts - 3 23
Imagen 18: Diagrama de clases del paquete charts - 4 24
Imagen 19: Diagrama de clases del paquete charts - 5 25
Imagen 20: Diagrama de clases del paquete charts - 6 26
Imagen 21: Diagrama de clases del paquete charts - 7 27
Imagen 22: Diagrama de clases del paquete charts - 8 28
Imagen 23: Diagrama de clases del paquete charts - 9 29
Imagen 24: Paso 1 Menú inicial 33
Imagen 25: Paso 2 Elección del menú de gráficas 34
Imagen 26: Paso 3 Elección de la BB.DD.. 34
Imagen 27: Paso 4 Elección del filtrado y de la tabla que contiene los datos 34
Imagen 28: Paso 5 Elección de las variables 35
Imagen 29: Paso 6-1 Creación de la métrica 35
Imagen 30: Paso 6-2 Creación de la métrica 35
Imagen 31: Paso 7 Elección del layout de la gráfica 36
Imagen 32: Paso 8 Elección del tamaño de la gráfica 36
Imagen 33: Resultado Ejemplo 37
Imagen 34: Gráfica con leyenda 37
Imagen 35: Gráfica con valores ocultos 38
Imagen 36: Dos gráficas con leyenda común 38
Imagen 37: Menú de configuración de la gráfica 39

Imagen 38: Gráfica personalizada 40
Imagen 39: Gráfica con ROLLUP 40
Imagen 40: Gráfica con CUBE 41

Índice de tablas

Tabla 1: Requisitos Funcionales 11
Tabla 2: Requisitos No Funcionales 11

Resumen

Cada día se recopilan más datos en todas partes, cada vez que entras en una página web o realizas una compra en una tienda o le das al *like* a una publicación de una empresa en Facebook se recopilan datos para su posterior análisis.
El uso de gráficas facilita mucho el análisis de los datos y la toma de decisiones en base a ellos, por eso se propuso desarrollar una interfaz web para la creación de varios tipos de gráficas y diferentes parámetros de configuración para cada tipo de gráfica. Con la interfaz web se elimina la necesidad de instalar programas por parte del usuario ya que cualquier ordenador cuenta con un navegador web.

Palabras clave

Interfaz web, análisis de datos, gráficas, leyenda común, Vaadin

Summary

Everyday data is gathered everywhere, each time you visit a website or buy something in a shop or give a *like* to an enterprise publication on Facebook some data is gathered for a later analysis.
The use of graphics facilitates a lot the data analysis and taking decisions in base of that data, so that's why it was proposed to develop a web interface for the creation of different types of graphics and different configuration parameters for each type of graphic.
Using a web interface eliminates de need of installing software in the user's computer because every computer has a web browser by default.

Key words

Web interface, data analysis, graphics, shared legend, Vaadin

1. Introducción

1.1. Introducción y contexto del proyecto

Hoy día se manejan grandes cantidades de datos a la hora de tomar decisiones de cualquier tipo.
Esto puede ser una gran herramienta para la toma de decisiones, ya que se tienen grandes cantidades de datos, pero a la vez es un problema ya que al tener tantos datos es más difícil analizarlos.
Debido a que no se tiene mucho tiempo para tomar estas decisiones lo mejor es encontrar mecanismos para agilizarlas. Como dice el dicho "una imagen vale más que mil palabras", para el análisis de información a menudo se recurre a la utilización de gráficas, con las que con un simple vistazo puedes darte cuenta de la magnitud de los datos que a simple vista parecían no tener, también ayuda a la hora de explicar los datos a otras personas que pueden no estar muy al tanto de la materia, y de paso obtienes una presentación de los datos más vistosa.

1.2. Motivación del trabajo a realizar

Debido a este problema, desde la empresa Predictia se planteó la idea de desarrollar una interfaz web donde poder crear distintos tipos de gráficas con datos que extraemos de una base de datos con altas hospitalarias para evitar el tener que introducirlos manualmente.
La decisión de realizar una interfaz web y no de escritorio es debido a que todos los ordenadores vienen normalmente con un navegador preinstalado con el sistema operativo y a la comodidad de no tener que instalar ningún programa para poder utilizarlo.
Como se manejan muchos datos a la vez con una sola gráfica no podemos tener una idea de diferentes variables a la vez, por lo que lo mejor es poder tener en pantalla varias gráficas al mismo tiempo.
Este trabajo toma como partida el que desarrollé durante las prácticas que realicé en Predictia.

Para la creación de gráficas utilizaremos datos multidimensionales, por lo que los datos se dividirán en dimensiones (o variables de desagregación) y medidas (o métricas).
En función de estos tipos de datos se crearán variables categóricas como el sexo (hombre o mujer) o el tipo de intervención, y numéricas, como la edad o el código postal. Con esos datos se usarán distintas métricas para analizarlos como conteos, porcentajes o medias.

En función del tipo de variable es más adecuado un tipo de gráfica u otro, por eso se desarrollarán varios tipos de gráficas para adecuarse a las necesidades de los datos. Para las primeras, podría utilizarse una gráfica tipo tarta ya que no suelen tener muchos valores y ayudan a expresar la repartición frente al total, para las segundas, por ejemplo, una gráfica de barras o líneas ya que al haber más valores se ve mejor. En el caso de ser una variable temporal, lo mejor es usar una gráfica temporal, que muestra la evolución de la variable respecto al tiempo.

Gracias a la librería que desarrolló Fernando Magaldi en un Proyecto de Fin de Carrera anterior en la misma empresa, se pueden codificar las variables de desagregación y se proporciona la información que representa dichas variables más allá del código almacenado en la BB.DD.

1.3. Objetivos

La realización de la interfaz para el análisis de datos contará con los siguientes objetivos:

- Diseñar e implementar varios tipos de gráficas con diferentes variables.
- Diseñar y añadir la posibilidad de realizar sumarizaciones de los datos, como roll-up y cube, pero sólo en los tipos de gráficas en las que su utilización sea coherente.
- Las gráficas tendrán varias opciones de personalización tales como el color de los datos o los títulos de las gráficas.
- Diseñar e implementar la posibilidad de crear varias gráficas y visualizarlas al mismo tiempo para analizar más información a la vez o compararla entre ella, además se podrá eliminar la gráfica que se desee manteniendo las demás.
- Diseñar e implementar una forma de controlar que las leyendas que acompañan a las gráficas se gestionen de modo que al haber una variable común en varias gráficas, ésta desaparezca de la leyenda individual de cada gráfica y aparezca en una leyenda centralizada desde la que se controlarán todos las gráficas, que la contengan, al mismo tiempo.
- Diseñar e implementar varias formas de presentar las gráficas, en concreto una basada en un *grid* con varias gráficas por fila y otra en al que se pueden reordenar las gráficas a nuestro gusto mediante la técnica de *drag&drop* arrastrar y soltar la gráfica hasta la posición deseada.

2. Material y métodos utilizados

2.1. Metodología

Para la realización de este proyecto y tras comparar las distintas metodologías más utilizadas decidí utilizar una metodología iterativa.

En un desarrollo iterativo e incremental el proyecto se planifica en diversos bloques temporales llamados iteraciones.
Las iteraciones se pueden entender como mini proyectos: en todas las iteraciones se repite un proceso de trabajo similar (de ahí el nombre "iterativo") para proporcionar un resultado completo sobre *producto final*, de manera que el cliente pueda obtener los beneficios del proyecto de forma incremental. Para ello, cada requisito se debe completar en una única iteración: el equipo debe realizar todas las tareas necesarias para completarlo (incluyendo pruebas y documentación) y que esté preparado para ser entregado al cliente con el mínimo esfuerzo necesario. De esta manera no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos.
En cada iteración el equipo evoluciona el producto (hace una entrega incremental) a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos/requisitos o mejorando los que ya fueron completados. Un aspecto fundamental para guiar el desarrollo iterativo e incremental es la priorización de los objetivos/requisitos en función del valor que aportan al cliente**.** (1)

Primero diseñé la estructura básica de la aplicación, las gráficas, desarrollándolo y más tarde probándolo. Una vez que la estructura básica estaba implementada y era estable, la amplié con diferentes funcionalidades como poder poner múltiples gráficas a la vez o personalizarlas, hasta llegar a la aplicación totalmente desarrollada.

En mi opinión, es una forma de desarrollar el proyecto más adecuada que una metodología en cascada ya que a medida que se va cogiendo soltura en el desarrollo de la aplicación se va teniendo una aplicación usable con la que poder ver si las ideas que tenías al principio son factibles o hay que enfocarlas de otra manera. De la otra forma hasta el final no empiezas a desarrollar el programa y te puedes dar de bruces contra un muro insalvable.

2.2. Planificación del trabajo

El trabajo ha comenzado con un estudio inicial del *framework* utilizado durante el desarrollo para poder comprender como funciona y saber aprovecharlo.

Tras diseñar la aplicación y decidir qué elementos se incluirían, se ha realizado la planificación que se puede ver el siguiente diagrama.

Imagen 1: Planificación del trabajo

Primero se realizarán las funcionalidades básicas:

- La posibilidad de tener varias gráficas a la vez en pantalla.
- Poder eliminar gráficas que ya no nos interesan.
- Poder realizar *scroll* en la leyenda de la gráfica cuando hay muchas variables.

Posteriormente se implementarán las opciones adicionales que mejoran es desarrollo inicial:

- Poder personalizar las gráficas.
- Controlar que valores se podían personalizar dependiendo de la gráfica.
- Poder reordenar las gráficas.
- Poder elegir el tamaño con el que se crearán las gráficas.
- Permitir ocultar las leyendas.
- Leyenda común cuando hay alguna variable en varias gráficas.

Aunque en la metodología iterativa se establecen bloques temporales, en este caso no ha habido una planificación temporal propiamente dicha, cuando se acaba de implementar y de probar una iteración, se pasa a la siguiente y así hasta el final del desarrollo.

2.3. Tecnologías, lenguajes y herramientas

A continuación expondré las tecnologías, lenguajes y herramientas utilizados para el diseño, implementación y explotación del trabajo realizado.

2.3.1. JAVA

El lenguaje de programación Java es un lenguaje de alto nivel, orientado a objetos. El lenguaje es inusual porque los programas Java son tanto compilados como interpretados. La compilación, que ocurre una vez por programa, traduce el código Java a un lenguaje intermedio llamado Java bytecode. El Bytecode, por su turno, es analizado y ejecutado (interpretado) por Java Virtual Machine (JVM) - un traductor entre el idioma, el sistema operativo subyacente y el hardware. Todas las implementaciones del lenguaje de programación deben emular JVM, para permitir que los programas Java ejecuten en cualquier sistema que tenga una versión de JVM.

La plataforma Java es una plataforma sólo de software que ejecuta sobre la base de varias plataformas de hardware. Está compuesto por JVM y la interfaz de programación de aplicaciones (API) Java un amplio conjunto de componentes de software (clases) listos para usar que facilitan el desarrollo y despliegue de applets y aplicaciones. La API Java abarca desde objetos básicos a conexión en red, seguridad, generación de XML y servicios web. Está agrupada en bibliotecas conocidas como paquetes de clases e interfaces relacionadas.

Además de la API JAVA, toda implementación completa de la plataforma Java incluye:

- Herramientas de desarrollo para compilar, ejecutar, supervisar, depurar y documentar aplicaciones.
- Mecanismos estándar para desplegar aplicaciones para los usuarios.
- Kits de herramientas de interfaz de usuario que permiten crear interfaces gráficas de usuario (GUIs) sofisticadas.
- Bibliotecas de integración que permiten que los programas accedan a bases de datos y manipulen objetos remotos. (2)

Aunque la aplicación se basa en una interfaz web, ésta ha sido desarrollada en JAVA. Esto es debido a que para el desarrollo de la aplicación se ha utilizado un framework llamado Vaadin el cual se encarga de transformar el lenguaje JAVA a HTML y JavaScript. Como casi toda la programación que realizamos durante la carrera se realiza en JAVA nos pareció lo más cómodo el uso de este framework.

2.3.2. Vaadin

Vaadin es un *framework* open source para el desarrollo de aplicaciones web modernas o también llamadas RIA (Rich Internet Applications) que posee una librería con numerosos componentes de usuario con los que construir aplicaciones web profesionales similares a los interfaces de una aplicación de escritorio. Este framework se ejecuta del lado del servidor lo que significa que la mayoría de la lógica y por tanto la mayor carga del trabajo recae en el servidor.

Vaadin utiliza tecnología Ajax del lado del cliente cuya base es GWT (Google Web Toolkit) para renderizar la interfaz de usuario en el navegador. Las aplicaciones Google Web Toolkit se escriben en Java y son automáticamente "traducidas" a JavaScript y HTML. GWT es muy cómodo para el desarrollador porque le mantiene al margen de estas acciones de presentación y sin la necesidad de instalar ningún *plugin* para su presentación. (3)

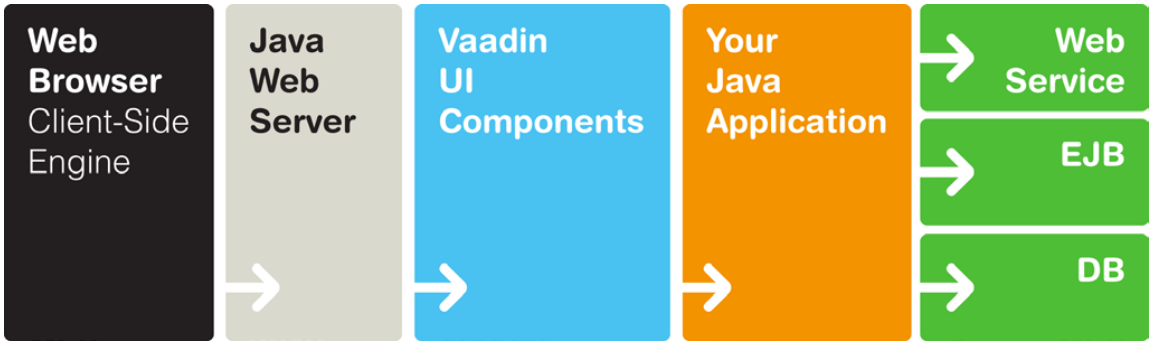

Imagen 2: Arquitectura cliente-servidor de Vaadin

Es open source, por lo que puedes utilizarlo y modificarlo a tu gusto y es compatible con los *plugins* desarrollados por los usuarios o los propios desarrolladores del *framework*. Si quieres una funcionalidad que no viene por defecto puedes buscarla que seguro que algún usuario la habrá creado y la puedes utilizar sin ningún problema.
Es muy fácil de usar, descargas el *plugin* para tu IDE favorito o puedes utilizar Maven, que es un gestor de dependencias, y ya puedes comenzar a utilizar sus funcionalidades como si de una librería más se tratase.

2.3.3. HighCharts

Las gráficas utilizadas en el *framework* Vaadin se apoyan en HighCharts, un motor de visualización de gráficas de la empresa HighSoft.

HighCharts es uno de los frameworks más conocidos el sector. Entre sus características principales destacan:

1. **Javascript:** Esta creado con JavaScript que es uno de los lenguajes más utilizados por parte de los desarrolladores.
2. **Compatibilidad:** La compatibilidad es **máxima** ya que soporta incluso Explorer 6 adecuándose a todos los navegadores y soportando un fuerte **cross browsing.**
3. **Tipos de Gráficas:** Otra de las cosas que más destaca de HighChart es el amplio soporte de gráficas de todo tipo que tienen permitiéndonos abordar la mayor parte de las casuísticas que nos encontremos.
4. **Facilidad de uso:** Por último destacar es que un framework muy sencillo de utilizar con el cual el desarrollador en poco tiempo podrá ser **productivo**. (4)

2.3.4. MySQL

Los datos utilizados en las gráficas se extraen de una base de datos relacional gestionada con MySQL.

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google (aunque no para búsquedas), Facebook, Twitter, Flickr, y YouTube. (5)

2.3.5. Librerías Predictia

Para el manejo de los datos extraídos de la base de datos en Predictia se ha desarrollado una librería con una serie de clases para transformar esos datos en objetos y poder trabajar con ellos de una manera más sencilla que con los datos en bruto extraídos directamente de la base de datos.

2.3.6. Predicados Guava

Para filtrar los datos obtenidos de la base de datos hemos utilizado los predicados de Guava, una librería de Google.

Google Guava es un conjunto de librerías comunes de código abierto para Java, desarrollado principalmente por ingenieros de Google.
Una parte de Guava son los predicados, con ellos se puede crear un método que determine si la entrada reúne o no las condiciones deseadas. Una vez tenemos creado el predicado, éste se puede usar como parámetro en un método de filtrado, así obtenemos solo los datos que queremos de todos los extraídos de la base de datos.

2.3.7. Subversion

Mi aplicación se engloba dentro de otra interfaz desarrollada paralelamente por otra persona, por lo que al trabajar concurrentemente utilizamos un gestor de versiones para poder trabajar sin conflictos de versiones entre nosotros.

Apache Subversion (abreviado frecuentemente como SVN, por el comando *svn*) es una herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache/BSD.

Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones (delta), optimizando así al máximo el uso de espacio en disco. SVN permite al usuario crear, copiar y borrar carpetas con la misma flexibilidad con la que lo haría si estuviese en su disco duro local. Dada su flexibilidad, es necesaria la aplicación de buenas prácticas para llevar a cabo una correcta gestión de las versiones del software generado.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada —si se ha hecho un cambio incorrecto a los datos, simplemente deshaga ese cambio. (6)

2.3.8. Tomcat

Para el despliegue de la aplicación y para su testeo hemos utilizado Tomcat.

Apache Tomcat, también conocido como simplemente Tomcat o Jakarta Tomcat, es un servidor web mutliplataforma que funciona como contenedor de servlets y que se desarrolla bajo el proyecto denominado Jackarta perteneciente a la Apache Software Foundation bajo la licencia Apache 2.0 y que implementa las especificaciones de los servlets y de JavaServer Pages o JSP de Sun Microsystem. Dicho servidor es mantenido y desarrollado por miembros de la fundación y voluntarios independientes, los cuales tienen libre acceso al código fuente bajo los términos establecidos por la Apache Software Foundation.

Apache Tomcat comenzó siendo una implementación de servlets iniciada por James Duncan Davidson, que trabajaba como arquitecto software en Sun Microsystem y que posteriormente ayudó al proyecto de código abierto. Duncan, inicialmente pensó que el proyecto se convertiría en software de código abierto y además quiso ponerle un nombre de animal, en este caso Tomcat (gato) ya que, de algún modo pretendía trasladar la posibilidad de cuidarse por sí mismo, es decir, de ser independiente. Las primeras distribuciones de Apache Tomcat fueron las 3.0.x aunque las versiones estables más recientes son las 6.0.30 y la 7.0.6 que implementan las especificaciones de Servlet 2.5 y JSP 2.1.

Como se ha dicho anteriormente, Apache Tomcat es un servidor web que da soporte a servlets y JSPs de modo que, no es un servidor de aplicaciones. Dado que dicho producto fue desarrollado en Java, éste puede ejecutarse sobre cualquier sistema operativo, previa instalación de la máquina virtual de Java aunque, también se puede usar con MAMPP (Mac OS X), LAMPP (GNU/Linux), WAMPP (Windows) o XAMPP (cualquier sistema operativo). Además, puede funcionar como servidor web por sí mismo sin embargo, en sus inicios se pensaba que dicho servidor era recomendable usarse en entornos de desarrollo con requisitos mínimos de velocidad. En la actualidad no existe esta percepción y por esto, es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Más en concreto, las principales características que soporta la última versión estable son:

- Autenticación de acceso básico.
- Negociación de credenciales.
- HTTPS.
- Alojamiento compartido.
- CGI o interfaz de entrada común.
- Servlets de Java.
- SSI.
- Consola de administrador. (7)

2.3.9. Eclipse

Como entorno de desarrollo se ha utilizado eclipse por ser el IDE utilizado tanto en Predictia como por mí durante la carrera.

Eclipse es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de plug-ins. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No tiene en mente un lenguaje específico, sino que es un IDE genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java usando el plug-in JDT que viene incluido en la distribución estándar del IDE.

Proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones.

Las principales características de Eclipse son:

- **Perspectivas, editores y vistas**: en Eclipse el concepto de trabajo está basado en las perspectivas, que no es otra cosa que una preconfiguración de ventanas y editores, relacionadas entre sí, y que nos permiten trabajar en un determinado entorno de trabajo de forma óptima.
- **Gestión de proyectos**: el desarrollo sobre Eclipse se basa en los proyectos, que son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios,... El IDE nos

proporcionará asistentes y ayudas para la creación de proyectos. Por ejemplo, cuando creamos uno, se abre la perspectiva adecuada al tipo de proyecto que estemos creando, con la colección de vistas, editores y ventanas preconfigurada por defecto.

- **Depurador de código**: se incluye un potente depurador, de uso fácil e intuitivo, y que visualmente nos ayuda a mejorar nuestro código. Para ello sólo debemos ejecutar el programa en modo depuración (con un simple botón). De nuevo, tenemos una perspectiva específica para la depuración de código, la perspectiva depuración, donde se muestra de forma ordenada toda la información necesaria para realizar dicha tarea.
- **Extensa colección de plug-ins**: están disponibles en una gran cantidad, unos publicados por Eclipse, otros por terceros. Al haber sido un estándar de facto durante tanto tiempo (no el único estándar, pero sí uno de ellos), la colección disponible es muy grande. Los hay gratuitos, de pago, bajo distintas licencias, pero casi para cualquier cosa que nos imaginemos tenemos el plug-in adecuado. (8)

3. Presentación del problema y análisis de requisitos

En esta aplicación sólo hay un tipo de usuario, por lo que hay un solo grupo de requisitos funcionales y de casos de uso.

3.1. Requisitos funcionales

Referencia	Requisito
RF01	El usuario podrá crear gráficas.
RF02	El usuario puede elegir la base de datos desde la que se obtendrá la información.
RF03	El usuario puede elegir filtrar o no los datos a extraer de la BB.DD..
RF04	El usuario puede elegir la tabla de la base de datos desde la que se obtendrá la información.
RF05	El usuario puede elegir si hay sumarización o no, y si la hay si es de tipo cube o rollup.
RF06	El usuario puede elegir las variables de la tabla que se usarán para la creación de la gráfica.
RF07	El usuario puede eliminar una variable una vez seleccionada.
RF08	El usuario puede elegir la métrica para la extracción de datos de la BB.DD., si es count, avg, etc…
RF09	El usuario puede modificar la métrica seleccionada.
RF10	El usuario puede eliminar la métrica seleccionada.
RF11	El usuario puede elegir el tipo de layout en el que se mostrarán las gráficas.
RF12	En el caso de haber elegido el layout “drag & drop”, el usuario podrá elegir el tamaño de las gráficas.
RF13	El usuario puede elegir el tipo de gráfica utilizada para mostrar los datos.
RF14	El usuario podrá configurar distintos parámetros de las gráficas como su altura, las líneas en las gráficas de tipo eje o los títulos y subtítulos de las gráficas.

RF15	En el caso de haber elegido el layout “drag & drop”, el usuario podrá mover las gráficas arrastrándolos de un lado a otro.
RF16	El usuario podrá eliminar una gráficas.
RF17	El usuario podrá mostrar/ocultar la leyenda de cada gráfica.
RF18	El usuario podrá ocultar/mostrar un valor de una variable de la gráfica clicando en la leyenda.

Tabla 1: Requisitos Funcionales

3.2. Requisitos no funcionales

Referencia	**Requisito**	**Categoría**
RNF01	La interfaz se debe visualizar correctamente en los navegadores Internet Explorer, Google Chrome y Mozilla Firefox.	Usabilidad
RNF02	Las gráficas se beberán generar en un tiempo < 10s, teniendo en cuenta que algunas variables tienen muchos valores.	Rendimiento
RNF03	La obtención de las variables de la base de datos solo se realizará la primera vez que se utilice la variable, las demás veces se consultarán en memoria.	Rendimiento
RNF04	No se dejarán huecos en blanco una vez eliminada una gráfica.	Usabilidad

Tabla 2: Requisitos No Funcionales

3.3. Diagrama de casos de uso

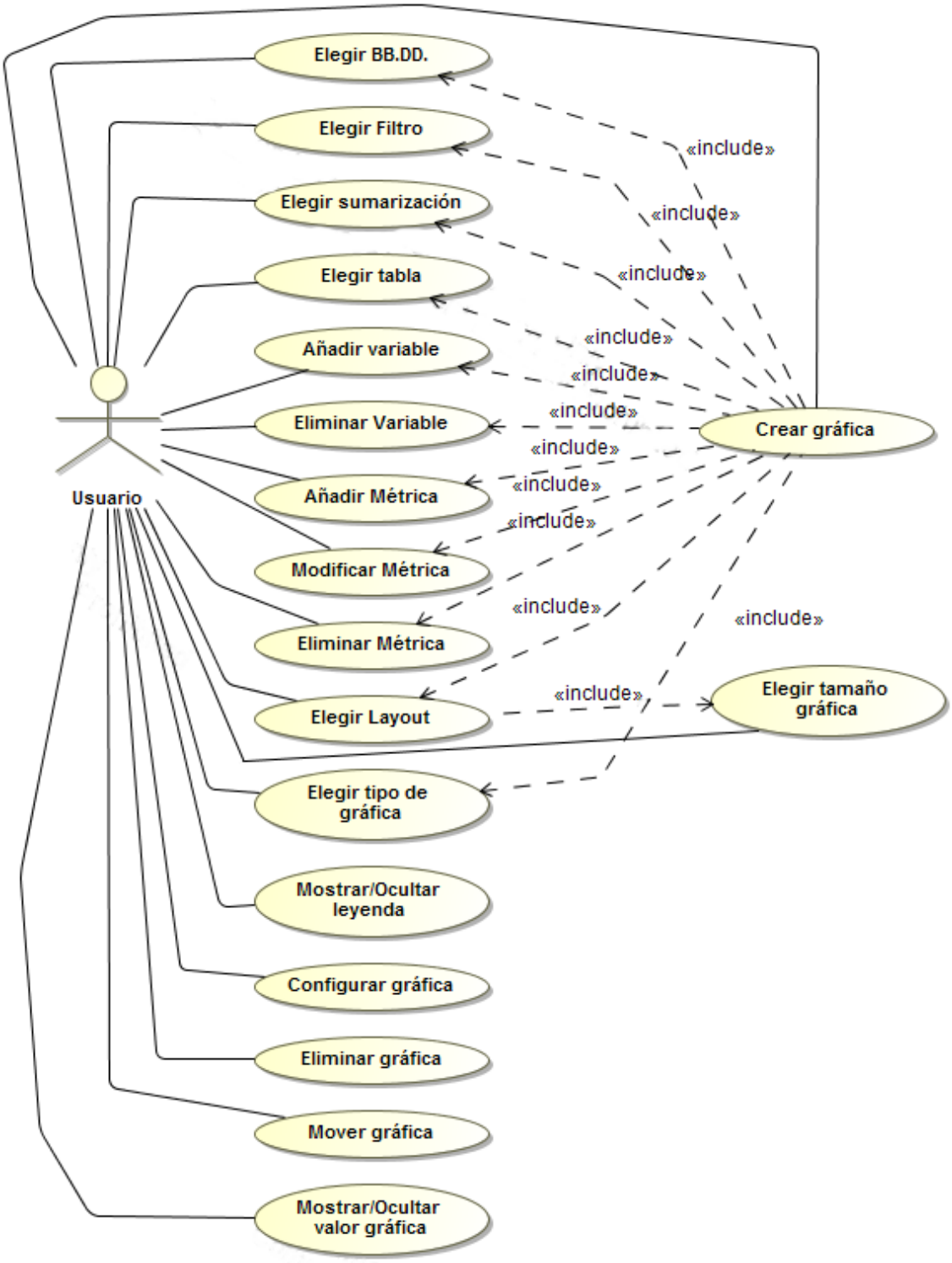

Imagen 3: Diagrama de Casos de Uso

4. Diseño e implementación

4.1. Diseño arquitectónico

Para el desarrollo de la aplicación ha utilizado un diseño en 3 capas, por lo que se divide en tres capas claramente diferenciadas:

- **Capa de presentación:** Es la capa con la que interactúa el usuario y se comunica con la capa de negocio. Gracias a Vaadin esta capa se desarrolla en lenguaje JAVA y es Vaadin el que se encarga de convertirlo a lenguaje HTML.
- **Capa de Negocio:** Es la que se encarga de la creación de las gráficas, de comunicarse con la capa de datos para extraer los datos y transformarlos en objetos java, y con la capa de presentación para mostrar las gráficas. Está desarrollada en JAVA.
- **Capa de persistencia:** Es la encargada de extraer los datos de la base de datos donde se encuentran almacenados. Al ser un aplicación de gráficas solo se realizan consultas. La base de datos está desarrollada en MySQL.

4.2. Diseño detallado e implementación

4.2.1. Capa de presentación

Esta capa es la que utiliza el usuario para interactuar con la aplicación, a medida que se van realizando los pasos para construir una gráfica se van desplegando las opciones, para que el usuario se centre en cada una y no le abrumen desde un primer momento.

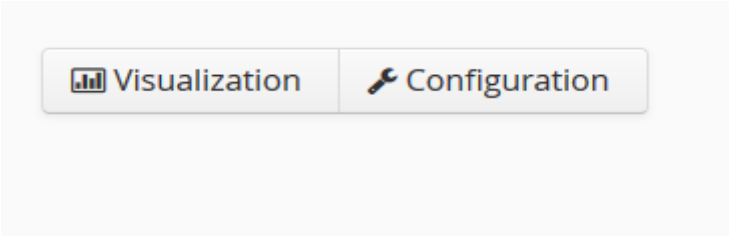

Imagen 4: Menú inicial

Como se podrá comprobar en el ejemplo práctico, para acceder a la interfaz de creación de gráficas se ha de seleccionar del menú inicial, que se puede observar en la imagen sobre estas líneas, la opción de Charts. Ese menú inicial y tanto la interfaz que se crea al elegir la opción Maps, como al hacer clic de Configuration, han sido desarrolladas por Fernando Magaldi Calleja en Predictia.

Como se observa en la imagen superior, al principio solo hay 2 botones y dependiendo del botón que cliquemos se generará un menú u otro, el menú se generará en base a las decisiones del usuario y no es fijo para todos los casos. Esto es debido a que la interfaz para la creación de gráficas está integrada dentro de una interfaz común y éste es el estado inicial de la interfaz común.

Como podemos ver en el código a continuación al clicar un botón se ejecuta el código de creación de otro menú y se añade al ya existente.

```java
public DathumMenu(ComponentContainer itemDestination) {
    super();
    this.componentUpdater = new ComponentUpdater(itemDestination, this);
    MenuItem viz = this.addItem("Visualization", FontAwesome.BAR_CHART_O, null);
    viz.addItem("Maps", FontAwesome.MAP_MARKER, updateWithComponent(new Supplier<Component>() {
        @Override
        public Component get() {
            return createFilterWithMapComponent();
        }
    }));
    viz.addItem("Charts", FontAwesome.BAR_CHART_O, updateWithComponent(new Supplier<Component>() {
        @Override
        public Component get() {
            return createChartComponent();
        }
    }));
    MenuItem config = this.addItem("Configuration", FontAwesome.WRENCH, null);
    config.addItem("DataDomain", FontAwesome.TABLE, updateWithComponent(new Supplier<Component>() {
        @Override
        public Component get() {
            return createDataDomainCrudComponent();
        }
    }));
    config.addItem("Variables", FontAwesome.COLUMNS, updateWithComponent(new Supplier<Component>() {
        @Override
        public Component get() {
            return createVariableCrudComponent();
        }
    }));
}

private MenuBar.Command updateWithComponent(final Supplier<Component> c){
    return new MenuBar.Command() {
        @Override
        public void menuSelected(MenuItem selectedItem) {
            component = Optional.of(c.get());
            componentUpdater.updateComponent();
        }
        private static final long serialVersionUID = -6696981225776424870L;
    };
}
```

Imagen 5: Creación del menú de la interfaz

Como en nuestro caso queremos crear gráficas, al clicar en el botón Charts centro del menú que se despliega al clicar en el botón Visualization, se ejecuta el código para crear el menú de una gráfica.

```java
public ChartComponent(DathumMessageSource messageSource, RepositoryHelper repositoryHelper, DataGroupService dataGroupService, FilterProvider...filterProviders) {
    super();
    ChartOptions.get().setTheme(new HighChartsDefaultTheme());
    this.messageSource = messageSource;
    this.repositoryHelper = repositoryHelper;
    this.dataGroupService = dataGroupService;
    this.extraFilters = filterProviders;
    addComponent(new DataDomainComboBox());
}

private class DataDomainComboBox extends ComponentUpdaterComboBox<DataDomain>{

    public DataDomainComboBox() {
        super(getChartComponent());
        setCaption(messageSource.getMessage("datadomain.select", "Data domain"));
        setItemCaptionFunction(CaptionBuilders.createDataDomainCaptionFunction());
        setAvailableElements(repositoryHelper.findAll(DataDomain.class));
        setNullSelectionAllowed(false);
    }

    @Override
    public Component generateComponentOnSelected(DataDomain dataDomain) {
        return new ChartGeneratorLayout(dataDomain);
    }

    private static final long serialVersionUID = -8279386811844599541L;
}
```

Imagen 6: Generación del menú de gráficas

Como podemos comprobar en el código de la imagen 5, se añade el menú para seleccionar la base de datos de la que se extraen los datos, al seleccionarla se genera el menú para la configuración de la creación de la gráfica y se añade al existente y así sucesivamente hasta mostrar toda la interfaz.

El desarrollo de la capa de presentación se divide en dos partes, la primera que es la que crea el primer menú, el de la imagen 4.

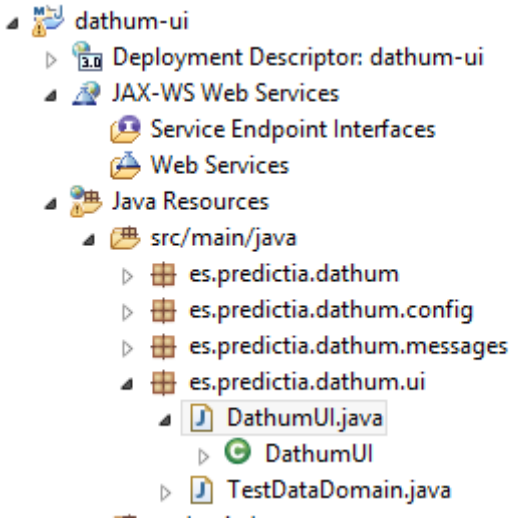

Imagen 7: Estructura código Primer menú

La segunda que se encuentra en otro proyecto aparte, el de las gráficas, por lo que son independientes, así si se quiere cambiar el menú de las gráficas solo hay que cambiar el tipo de componente que se crea al seleccionar la base de datos.

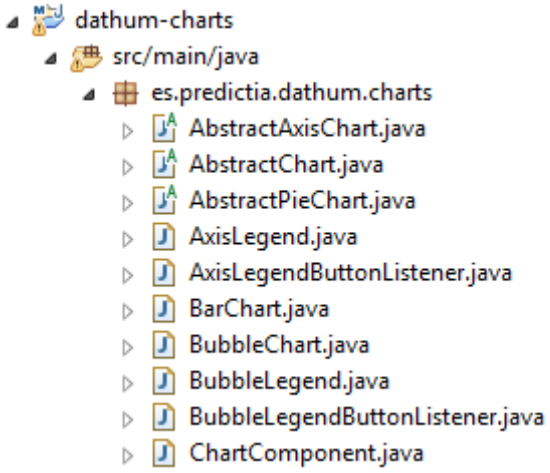

Imagen 8: Estructura código menú gráfica

4.2.2. Capa de negocio

La capa de negocio es la encargada de la creación de las gráficas y de su configuración. Esta desarrollada en java y se divide en varios paquetes.

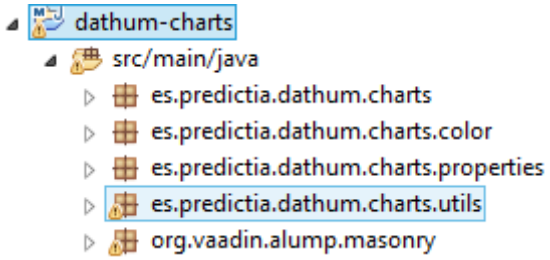

Imagen 9: Estructura Capa de negocio

Concretamente está dividido en:

1. **color**: Contiene los diferentes esquemas de colores que se pueden aplicar a las gráficas así como un generador que genera un numero diferente de colores para cada esquema en función del número de valores de la variable, así se intenta evitar la posibilidad de que haya colores repetidos la cual podría ser mayor si siempre se generaran un número fijo de colores.
2. **utils**: Contiene algunos métodos para el tratamiento de los datos extraídos de la base de datos y para el manejo de las leyendas.
3. **properties**: Contiene las diferentes propiedades con las que se pueden configurar los gráficos, como el tamaño de las burbujas para la gráfica de burbujas, si el fondo de la gráfica es blanco o transparente, los títulos y subtítulos de las gráficas, etc…
4. **charts**: Contiene los diferentes tipos de gráficas así como sus leyendas y los listeners de las leyendas, que son los encargados de actualizar las gráficas en función de las variables que se ocultan/muestran.

El paquete charts se puede dividir a su vez en dos grandes bloques:

1. **Gráficas y leyendas**
   1.1. **Ejes**: Las típicas gráficas que se usan casi siempre para mostrar datos. En este apartado se encuentran las gráficas de barras, de líneas y de barras apiladas. Las de barras y barras apiladas pueden configurarse para que sean tanto verticales como horizontales, por defecto vienen verticales.
   Las de líneas se pueden hacer tanto de líneas rectas, como curvas o de escalones
   1.2. **Burbujas**: Son ideales para expresar la magnitud de una variable, pero en vez de líneas, usa áreas para expresarlo. Necesitan dos variables para poder usarse ya que la intersección de las dos variables determina la posición de la burbuja y la métrica, el área.
   1.3. **Tarta**: Muestra adecuadamente la repartición de los datos con respecto al total. Se divide en dos, una gráfica de tarta normal y otra en la que se crea una gráfica individual por cada valor de la variable de agrupación, por si quieres ver la repartición de los datos pero individualizado por cada caso.
   1.4. **Temporal**: Muestran la evolución de una variable temporal, o de una variable con respecto a otra temporal.
   Cada tipo de gráfica tiene su leyenda creada específicamente para ella, debido a las particularidades de cada una, a excepción de la Temporal, para la que no se implementó. La leyenda común se colocará debajo del dashboard y engloba todas las variables comunes a más de una gráfica.
2. **Dashboards**
   2.1. ***Grid***: Las gráficas se colocan en una matriz con un número de gráficas por fila definido por el programador. A medida que se van añadiendo gráficas se van colocando en orden. Al eliminar una las demás se reordenan para ocupar el espacio que se dejó.
   2.2. ***Drag&Drop***: Las gráficas se van añadiendo según el hueco que hay disponible. Se pueden reordenar manualmente mediante la técnica de arrastrar y soltar. Se reordenan ocupando los huecos una vez se elimina una gráfica.

A continuación se muestra un diagrama de clases de la capa de Negocio. Se han omitido del diagrama los constructores y los getters y setters para reducir contenido.
En los diagramas muy grandes se han omitido los argumentos, sustituyéndolos por … en el caso de haberlos, para ahorrar espacio y se han dividido en varias imágenes para poder verlos correctamente.
Algunas clases está repetidas para mostrar las relacione entre ellas.

Paquete color

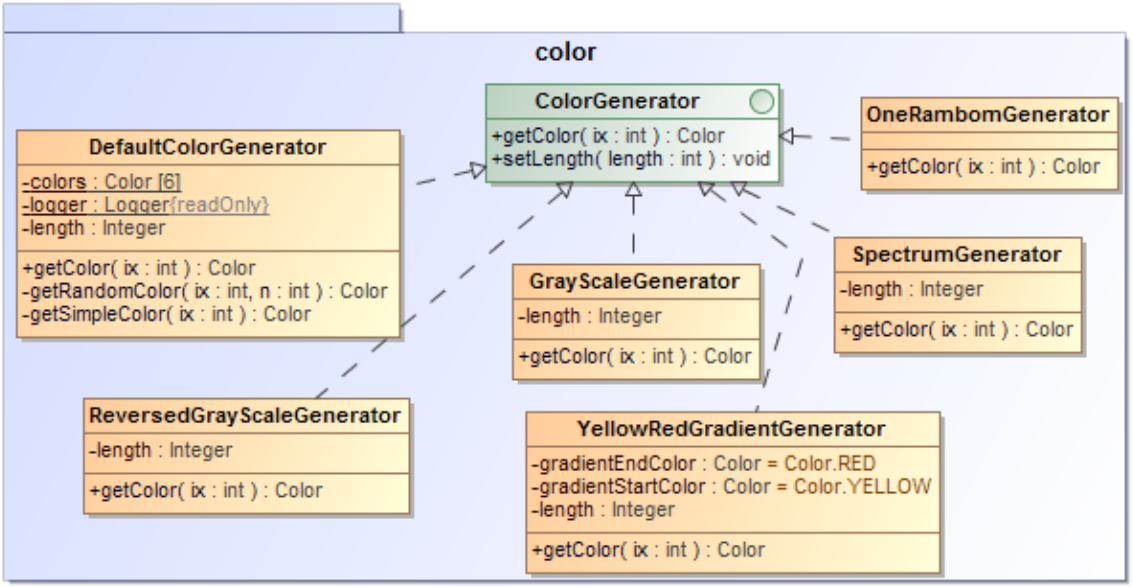

Imagen 10: Diagrama de clases del paquete color

Paquete utils

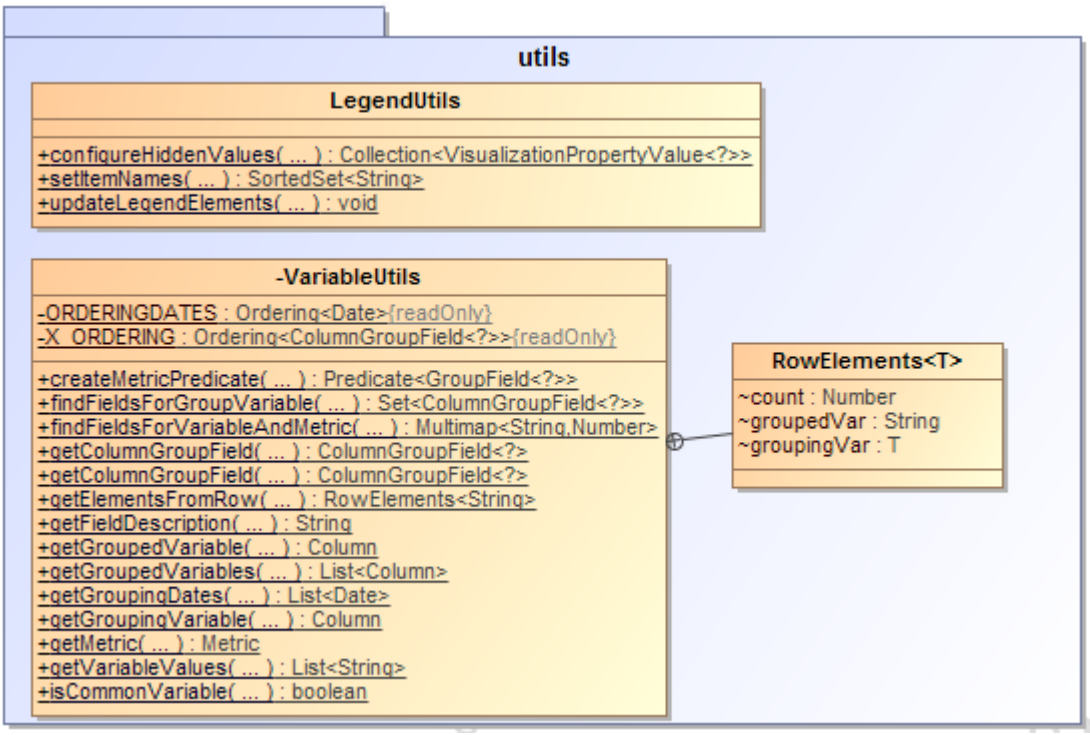

Imagen 11: Diagrama de clases del paquete utils

Paquete properties

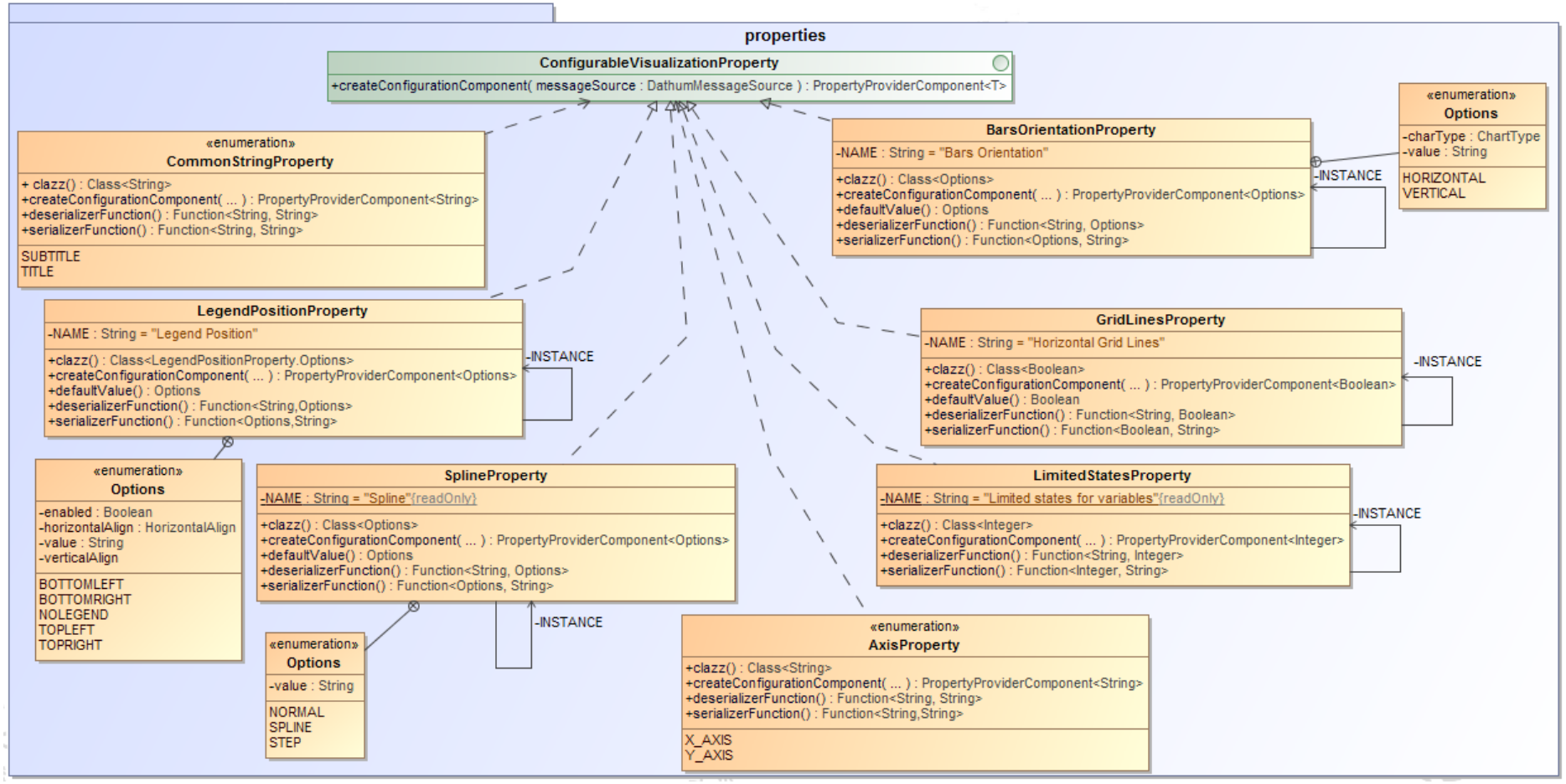

Imagen 12: Diagrama de clases del paquete properties - 1

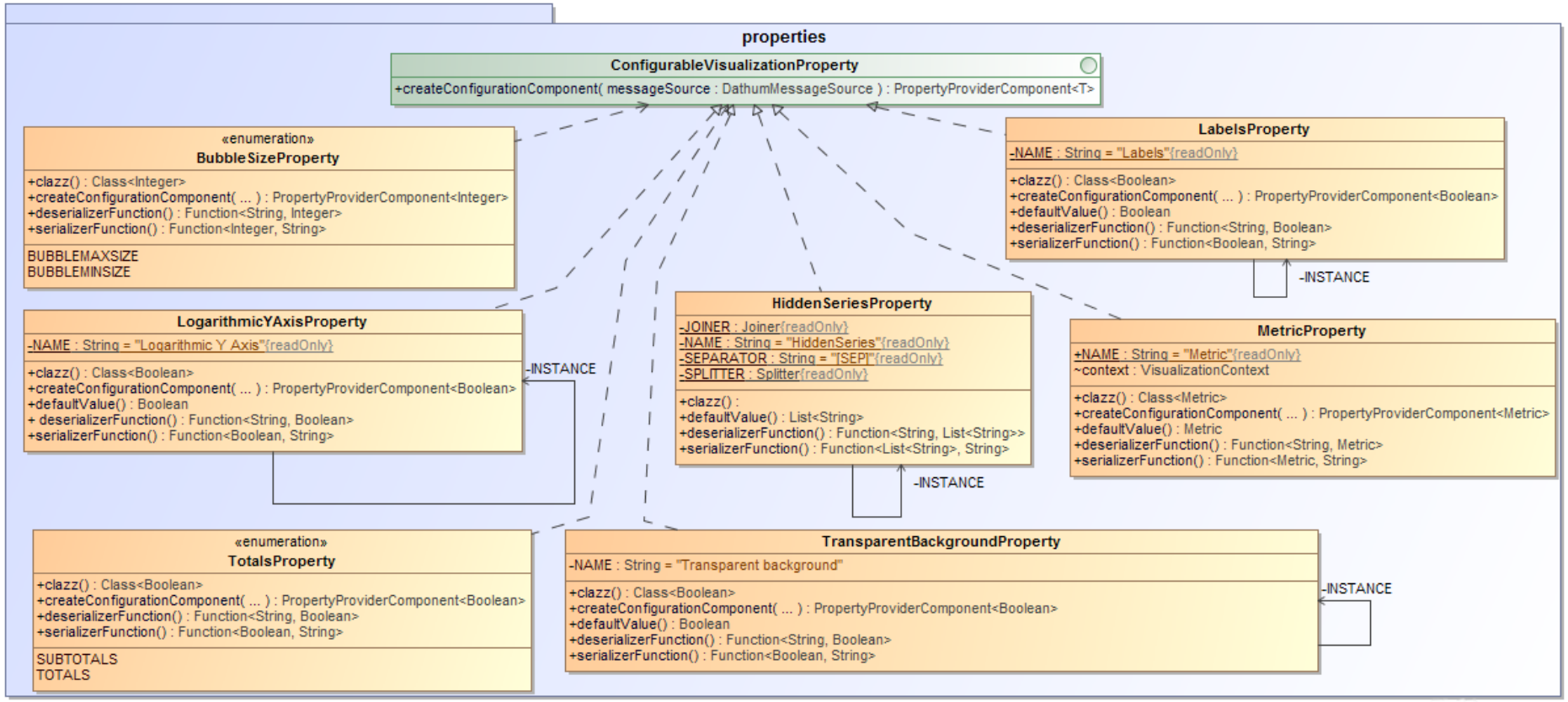

Imagen 13: Diagrama de clases del paquete properties - 2

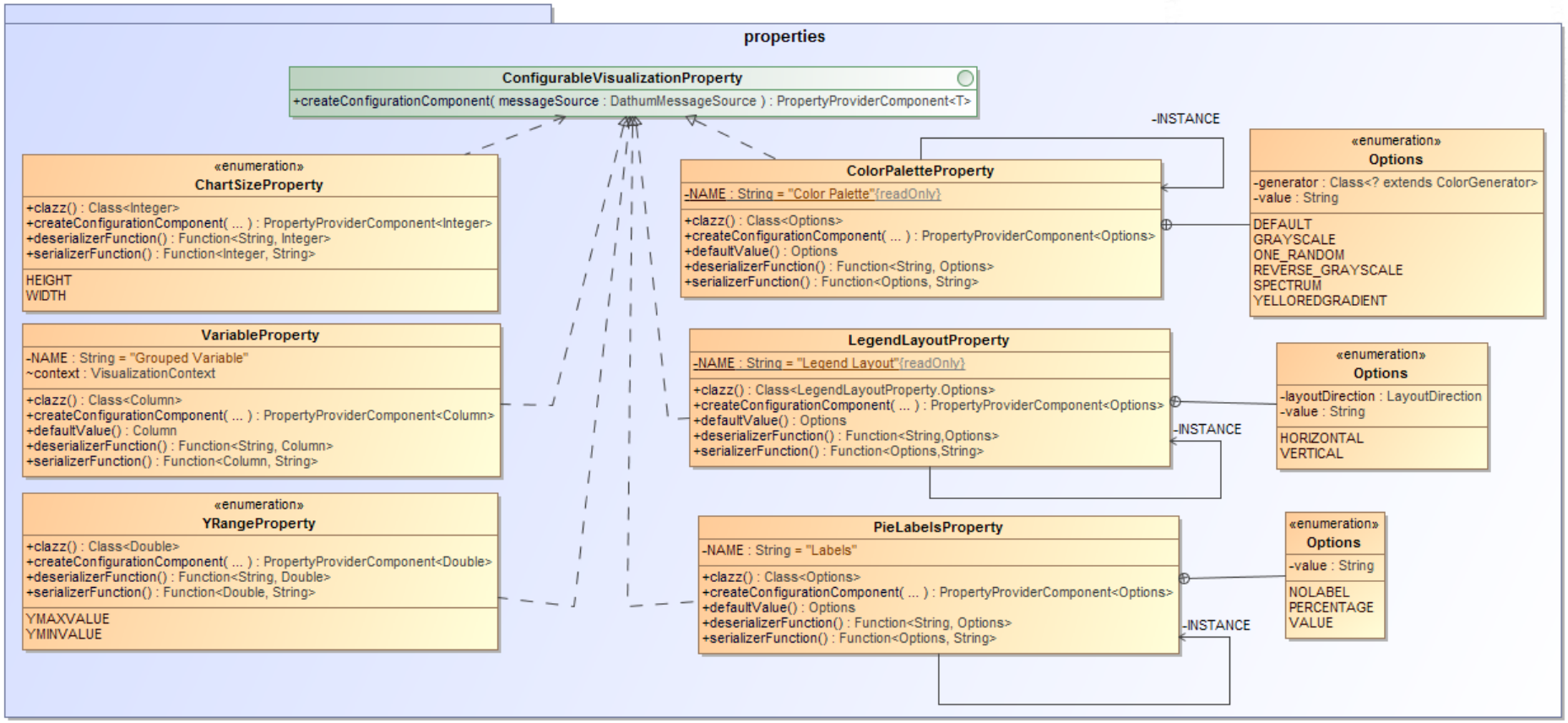

Imagen 14: Diagrama de clases del paquete properties - 3

Paquete charts

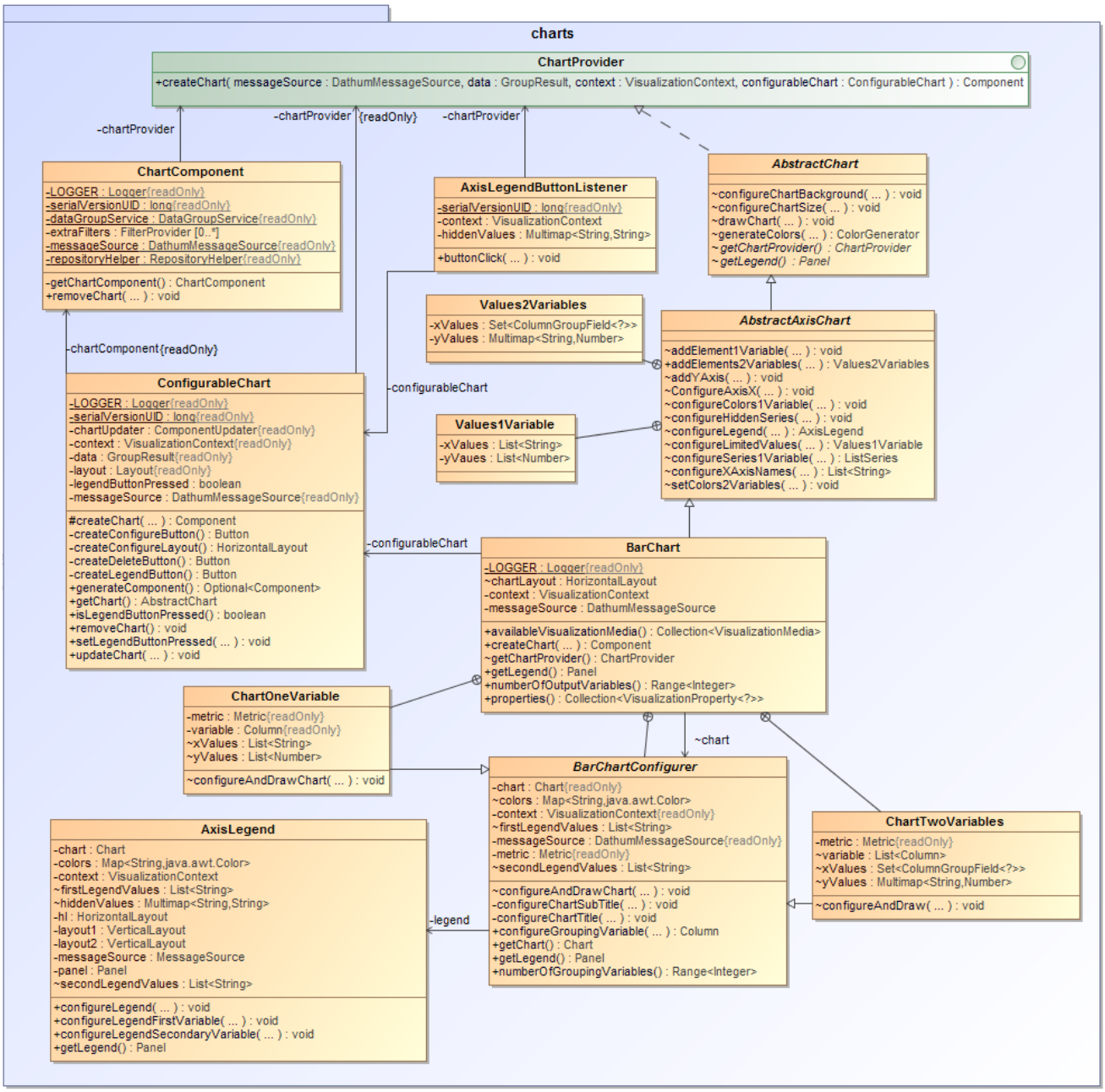

Imagen 15: Diagrama de clases del paquete charts - 1

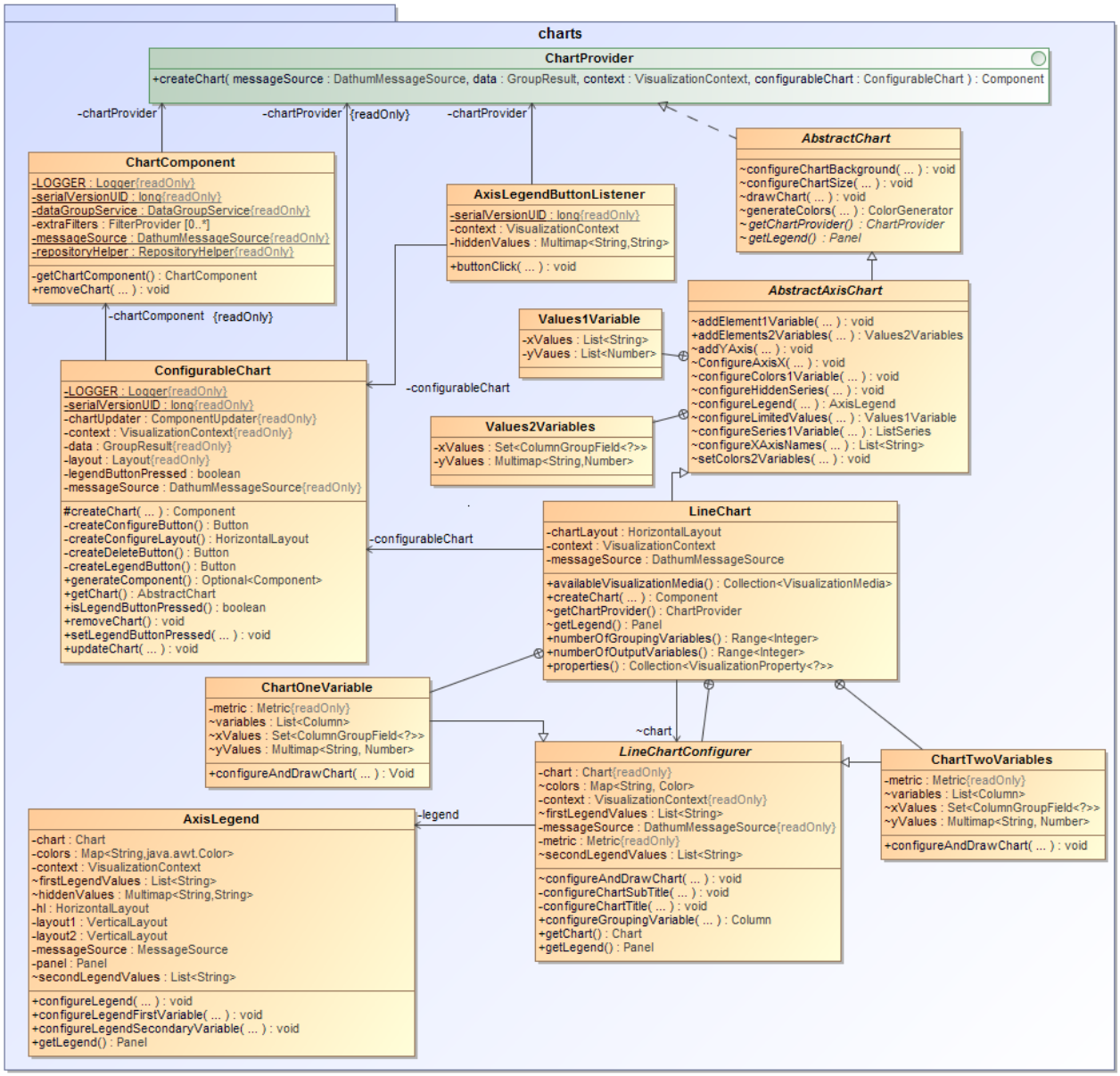

Imagen 16: Diagrama de clases del paquete charts - 2

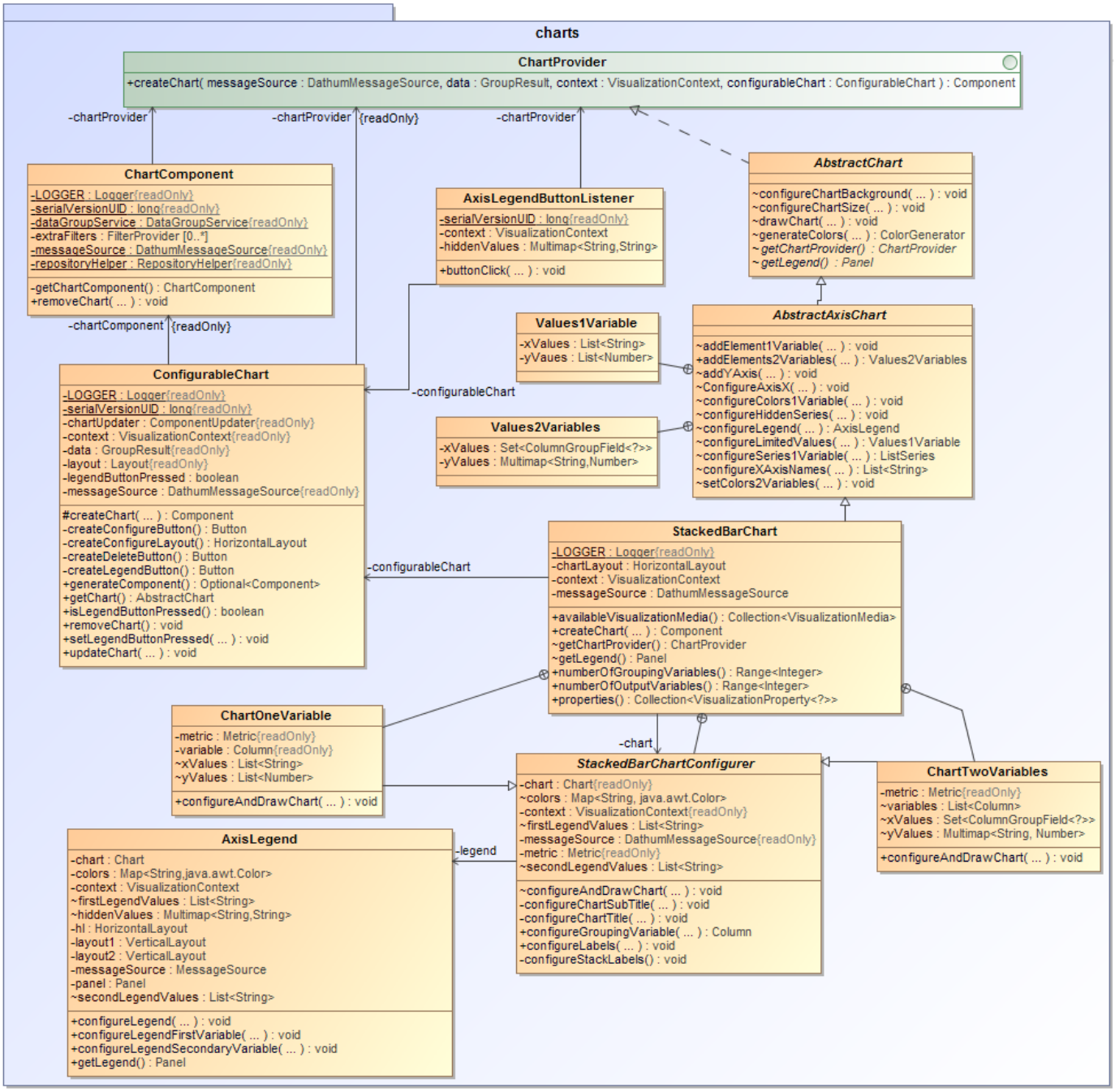

Imagen 17: Diagrama de clases del paquete charts - 3

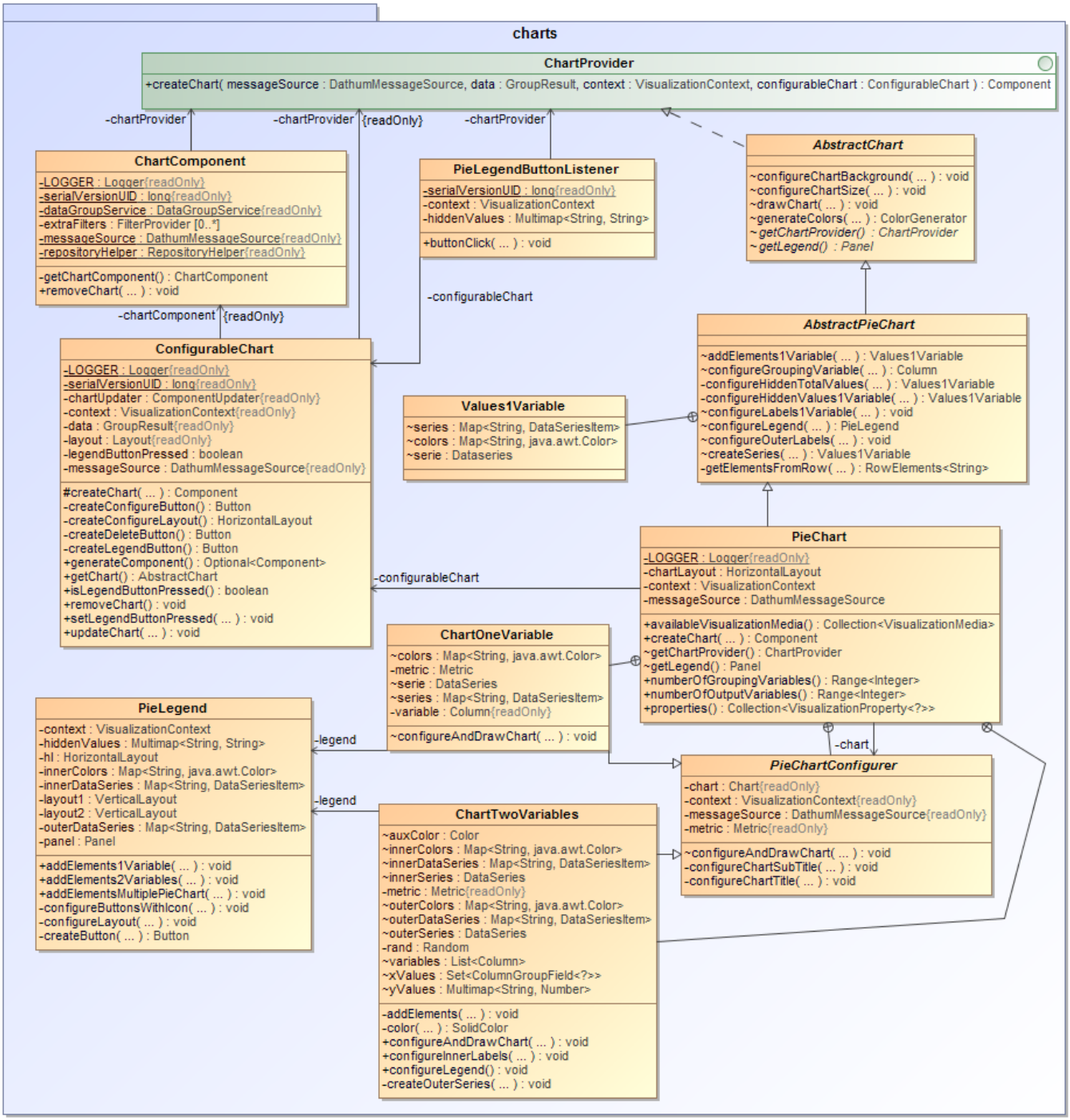

Imagen 18: Diagrama de clases del paquete charts - 4

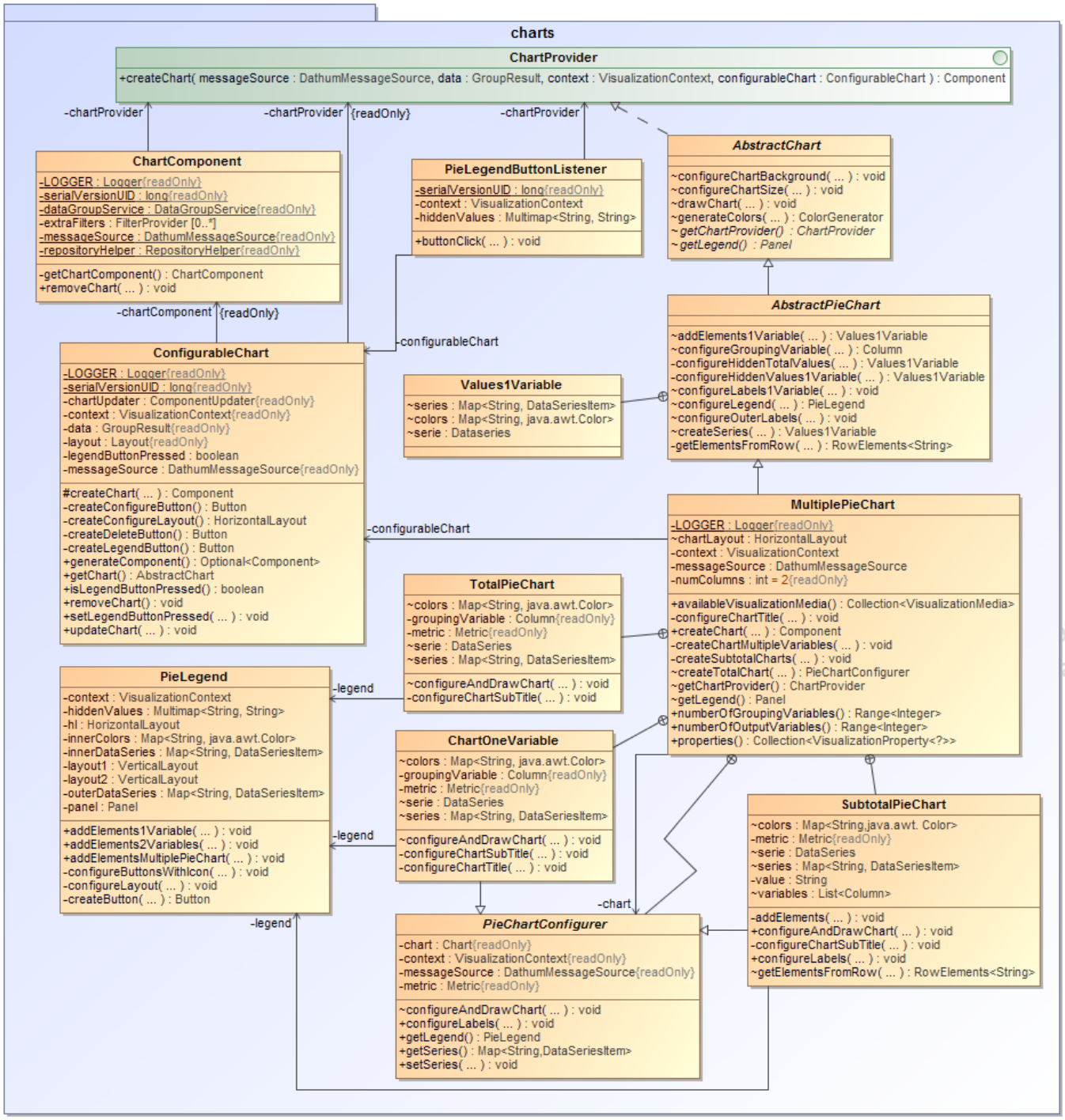

Imagen 19: Diagrama de clases del paquete charts - 5

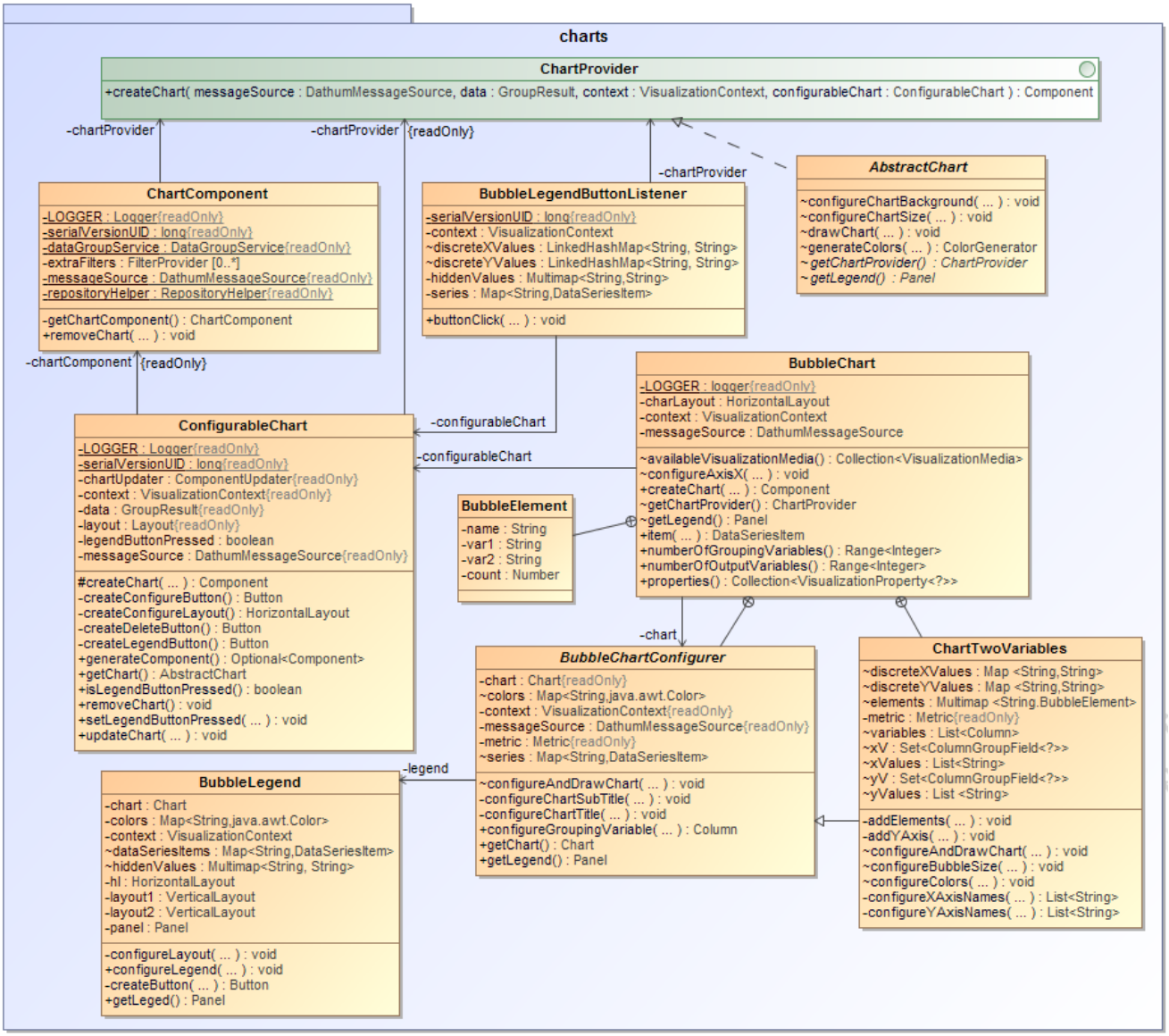

Imagen 20: Diagrama de clases del paquete charts - 6

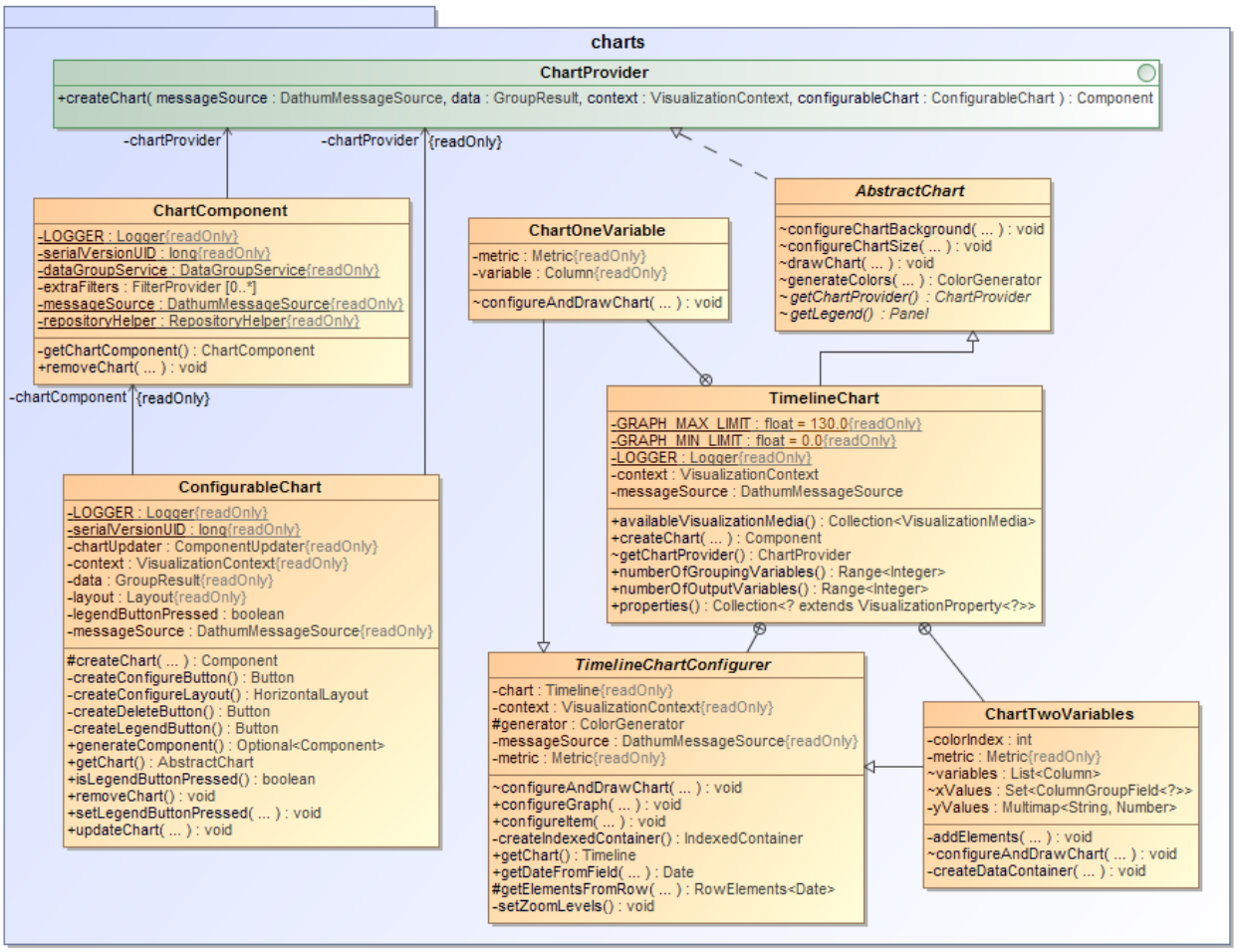

Imagen 21: Diagrama de clases del paquete charts - 7

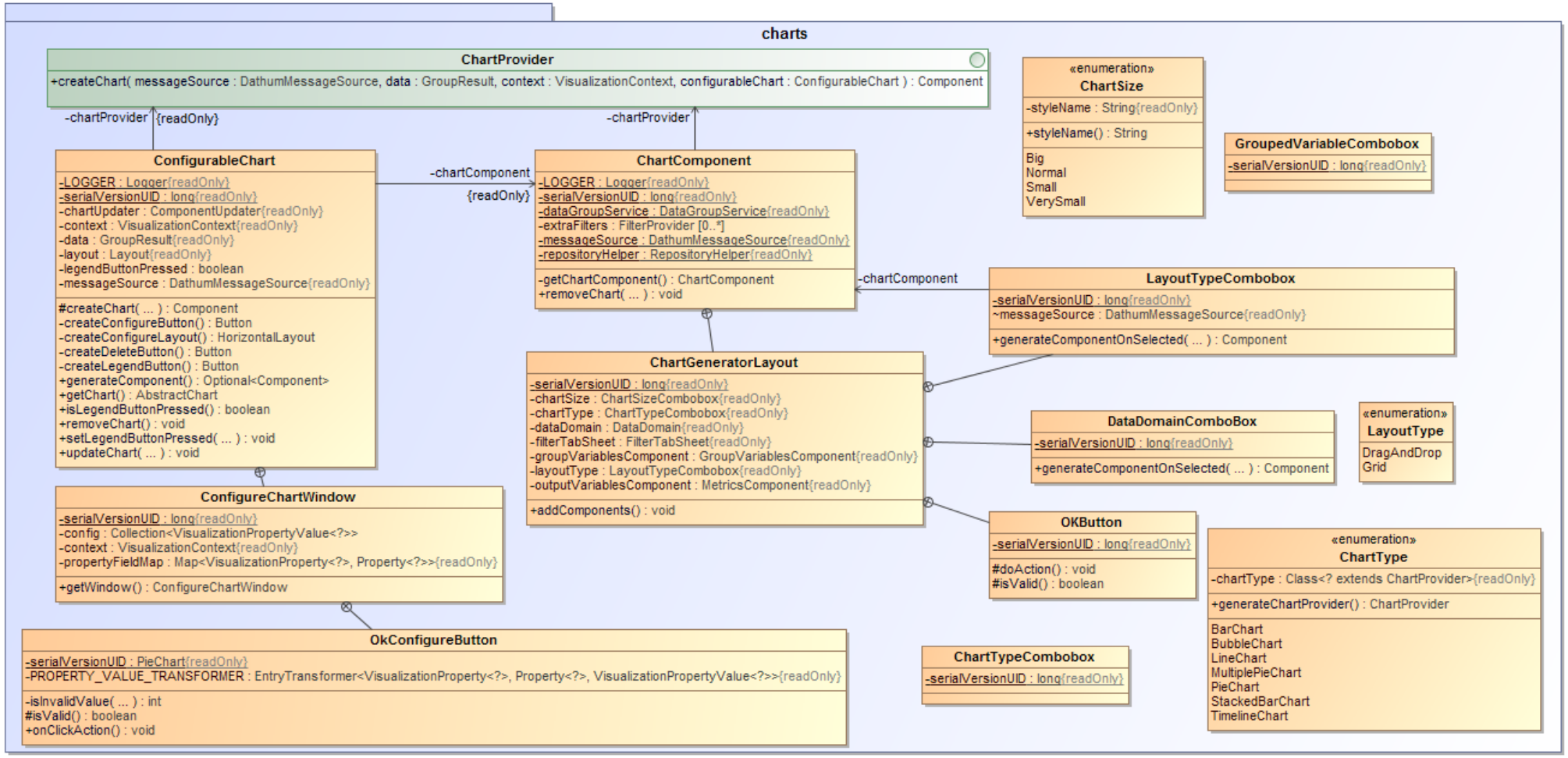

Imagen 22: Diagrama de clases del paquete charts - 8

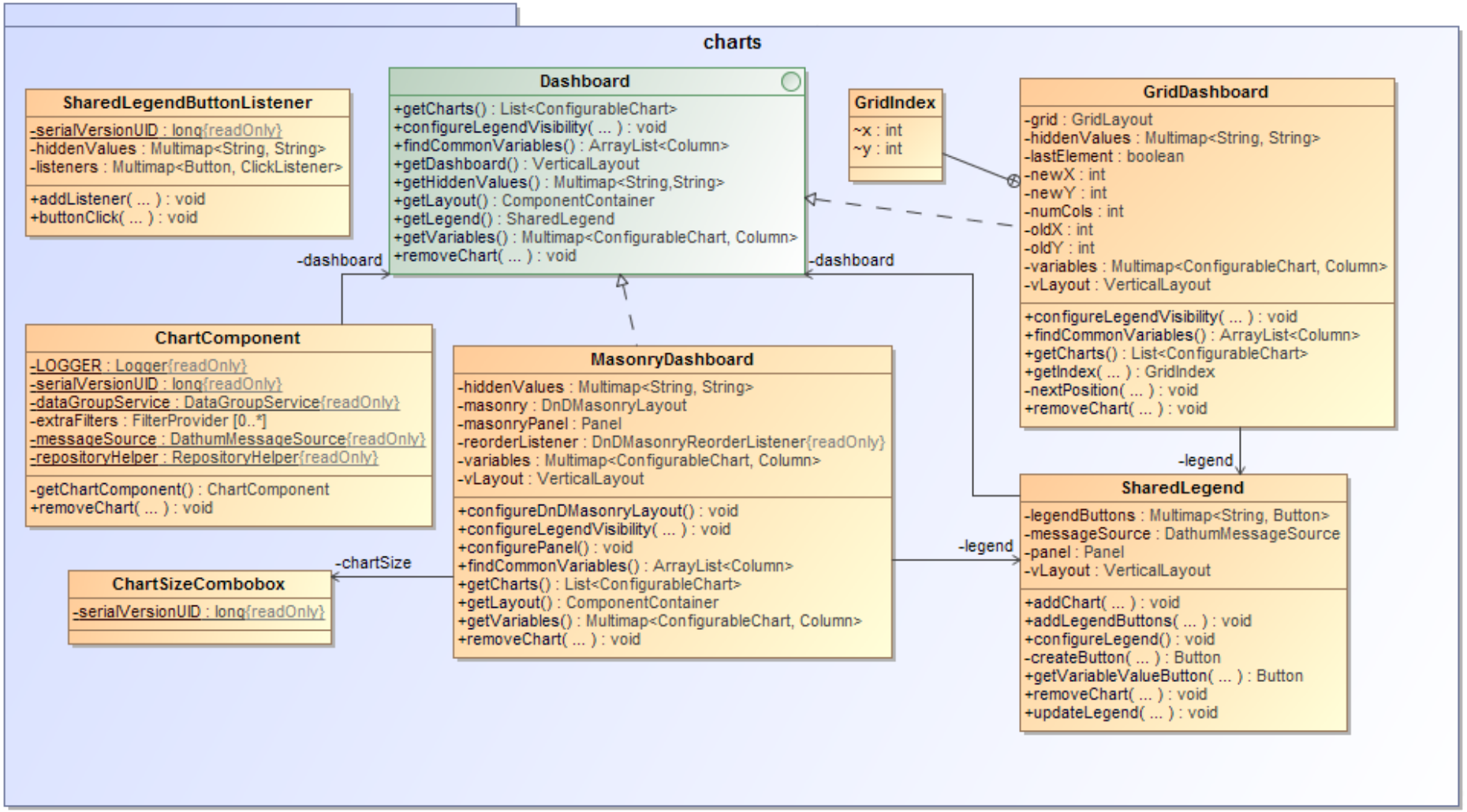

Imagen 23: Diagrama de clases del paquete charts - 9

4.2.3. Capa de persistencia

La capa de persistencia está formada por una base de datos mySQL donde se almacenan los diferentes datos médicos de cada paciente utilizando datos numéricos y fechas. Esta base de datos permite mostrar datos filtrados y agrupados.

Se almacenan datos como las fechas de ingreso y alta, los tipos de salida del hospital, los códigos de intervención o el sexo del paciente.
Algunos datos son discretos como le sexo o el tipo de alta, y solo pueden tomar unos valores determinados, otros son libres como la edad del paciente.

Los datos provenientes de la base de datos se transforman en objetos java para su posterior tratamiento en la capa de negocio.

Estas dos partes han sido desarrolladas por Predictia.

5. Pruebas

Debido a la complejidad de desarrollar pruebas automáticas para interfaces de usuario y debido a que la aplicación es una interfaz de usuario, desde Predictia optamos por realizar las pruebas a mano, como lo haría un usuario más utilizando la aplicación.
Como el desarrollo de la aplicación se ha realizado utilizando la metodología iterativa, las pruebas se han ido realizando a lo largo de todo el desarrollo.
Primero hemos empezado con las pruebas unitarias y a medida que íbamos desarrollando funcionalidades se realizaron las de integración. Una vez que teníamos la aplicación funcional se realizaron las pruebas de sistemas para comprobar la aplicación en su conjunto.

5.1. Pruebas unitarias

Tras desarrollar cada iteración se han realizado pruebas sobre ese módulo para comprobar su correcto funcionamiento.
A continuación expongo algunas pruebas tipo que se realizaron:

Prueba 1
Descripción:
Crear una gráfica de barras.

Resultado esperado:
La gráfica se muestra correctamente.

Forma de verificar:
Se siguen todos los pasos descritos en el punto 6 de la memoria, seleccionando el tipo de gráfica barchart.

Prueba 2
Descripción:
Intentar utilizar valores fuera de rango o erróneos en la configuración de la gráfica.

Resultado esperado:
Se muestra un mensaje indicando que la altura debe de ser mayor que 0.

Forma de verificar:
Se introduce un valor negativo en el campo Height del menú de configuración y se hace clic en OK.

Prueba 3
Descripción:
Ocultar un valor de una variable de la gráfica.

Resultado esperado:
Se ocultan de la gráfica las porciones para el valor hombre.

Forma de verificar:
Se crea una gráfica de tipo piechart con las variables sex y dischargeType, se creará una tarta con dos colores principales, uno por cada sexo, y un gradiente de ese color por cada valor de dischargeType para cada valor de sexo.
A continuación se clica en el botón Legend para que se visualice al leyenda.
Se hace clic en hombre.

Prueba 4
Descripción:
Ocultar un valor de una variable en varias gráficas.

Resultado esperado:
Se genera una legenda común y posteriormente se oculta el valor 4 de dischargeType de la gráfica tipo LineChart y de la BubbleChart.

Forma de verificar:
Se crean dos gráficas, una de tipo LineChart con las variables age y dischargeType, la otra de tipo BubbleChart con dos variables, dischargeType y sex.
Al crear la segunda gráfica aparece debajo de éstas una legenda con la variable dischargeType, al ser común a ambas. Se hace clic en el valor 4.

Prueba 5
Descripción:
Se reposicionan las gráficas al eliminar una.

Resultado esperado:
Al eliminar la gráfica, las siguientes se reposicionan para ocupar el hueco de la anterior y no dejar espacios en blanco.

Forma de verificar:
Se crean varias gráficas ya sea con el layout de tipo grid o el drag & drop.
Se elige una gráfica que no sea ni la última y se hace clic en el botón Delete.

Prueba 6
Descripción:
Reordenar una gráfica en layout drag & drop.

Resultado esperado:
La gráfica arrastrada ocupa la posición en la que se suelta.

Forma de verificar:
Se crean varias gráficas en un layout tipo drag & drop.
Se mantiene pulsado el botón izquierdo del ratón sobre la gráfica que se desea reubicar y sin soltarlo se arrastra hasta al posición deseada, una vez alcanzada se suelta el botón.

5.2. Pruebas de integración

A la vez que se han ido implementando nuevas funcionalidades, se han ido realizando pruebas con las funcionalidades implementadas anteriormente ya que de nada sirve que funcionen por separado si al trabajar juntas hay errores.

Por ejemplo se ha probado a realizar diferentes opciones de personalización a la vez para ver si podían coexistir sin generar errores.
Se ha probado a crear varias gráficas a la vez y probar como se comportaba la leyenda común y como respondían las gráficas al usarla.
Se han realizado diferentes combinaciones de funcionalidades también, como juntar varios tipos de gráficas y diferentes tipos de variables, temporales, discretas y normales, a la vez, y modificar la configuración de todas ellas.

5.3. Pruebas de sistema

Una vez teníamos la aplicación funcional se realizaron las pruebas de sistema para comprobar el correcto funcionamiento de todas las partes desarrolladas funcionando a la vez. Se realizaron pruebas utilizando todas las funcionalidades desarrolladas hasta ese momento. Cuando se implementó el layout con reordenación *drag&drop* o la leyenda común, se volvieron a realizar estas pruebas para comprobar que no se producían errores.

5.3.1. Pruebas de usabilidad

Se realizaron pruebas con diferentes navegadores para comprobar el correcto funcionamiento de la aplicación en todos ellos, se han utilizado los exploradores Google Chrome, que se ha usado como base de pruebas durante el desarrollo, y Mozilla Firefox, por ser los más extendidos.

Debido a que el desarrollo de la aplicación se hizo en un entorno Linux, no se hicieron pruebas sobre Internet Explorer hasta la finalización del desarrollo y su posterior exportación a mi equipo personal en el que he realizado las pruebas.

5.3.2. Pruebas de rendimiento

Se realizaron pruebas de rendimiento para comprobar si se tardaba un tiempo aceptable en la creación de las gráficas. Se observó que para grandes cantidades de datos, hablando de miles, o teniendo muchísimos valores diferentes en las variables, en el caso de las variables temporales, la aplicación se demora más de lo deseado, llegando a tardar incluso 6-7 segundos en el caso de tener miles de datos, pero para un uso normal las gráficas se generan en 1-2 segundos.

5.4. Pruebas de aceptación

Para las pruebas de aceptación se revisó que se hubiesen implementado todas las características que se planificaron y si lo hacían del modo en que se habían diseñado.
Por último se les pidió a las diferentes personas que trabajaban en Predictia que probaran la aplicación y que dieran su opinión de si les parecía que la aplicación funcionaba correctamente.

Más tarde también les di a probar la aplicación a mi familia ya que al ser personas ajenas al desarrollo de la aplicación entraban en contacto con la aplicación por primera vez, y les pedí que me diesen su opinión sobre si cumplía las funcionalidades que les dije que se había diseñado y si era fácil de usar.

6. Ejemplo Práctico

Creación de gráficas

A continuación se muestra la secuencia para crear una gráfica.

La interfaz forma parte de una aplicación más amplia que contempla el uso de mapas y la configuración de las variables. Nos vamos a centrar solo en los pasos necesarios para crear la gráfica.

Paso 1

La interfaz comienza con solo dos botones, para la visualización de gráficas, que es la parte que nos interesa se clica en el botón Visualization, por lo que se desplegará un menú para elegir que se quiere ver.

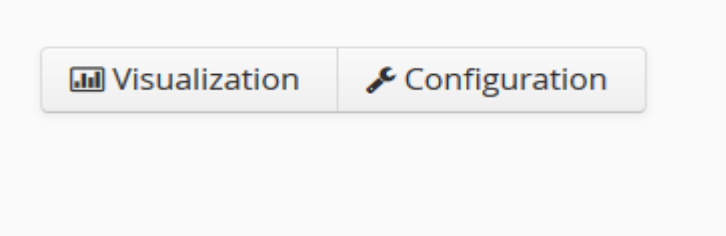

Imagen 24: Paso 1 Menú inicial

Paso 2

Elegimos la opción Charts, por lo que se desplegará la interfaz correspondiente a las gráficas.

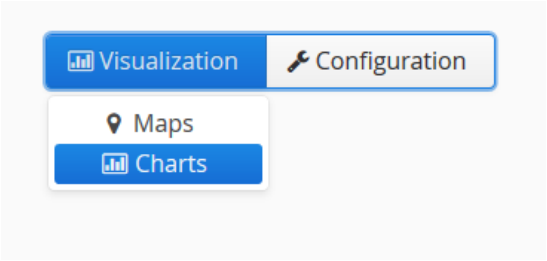

Imagen 25: Paso 2 Elección del menú de gráficas

Paso 3

Elegimos la base de datos desde la que se van a extraer los datos para la creación de la gráfica.

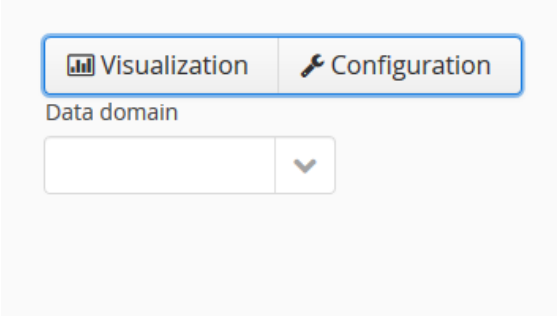

Imagen 26: Paso 3 Elección de la BB.DD.

Paso 4

Elegimos el tipo de filtrado de los datos y la tabla desde la que se extraerán los datos.

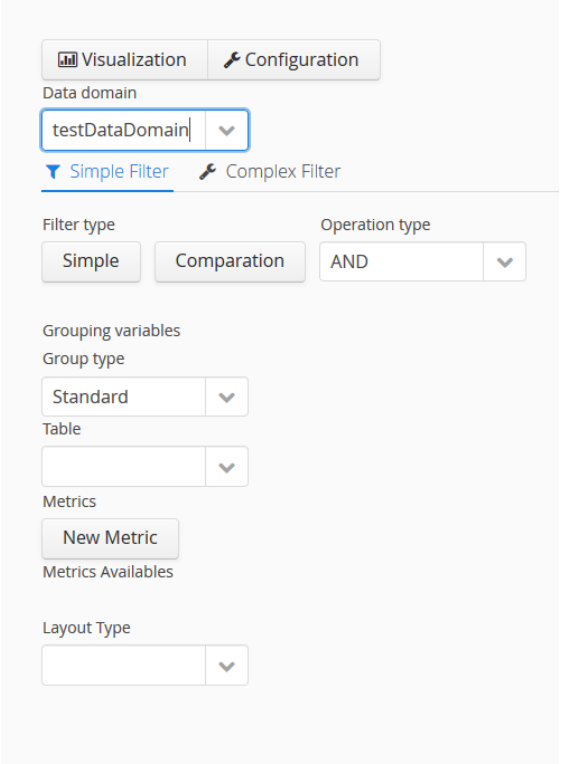

Imagen 27: Paso 4 Elección del filtrado y de la tabla que contiene los datos

Paso 5

Elegimos las variables que utilizaremos para la gráfica y el tipo de agrupamiento.

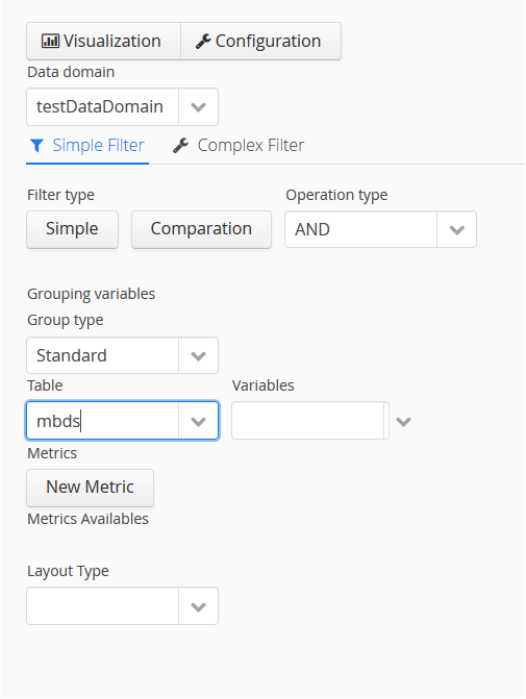

Imagen 28: Paso 5 Elección de las variables

Paso 6

Clicamos en New Metric y nos aparece una ventana donde elegimos el tipo de métrica que se va usar con los datos y le damos un nombre para crearla.

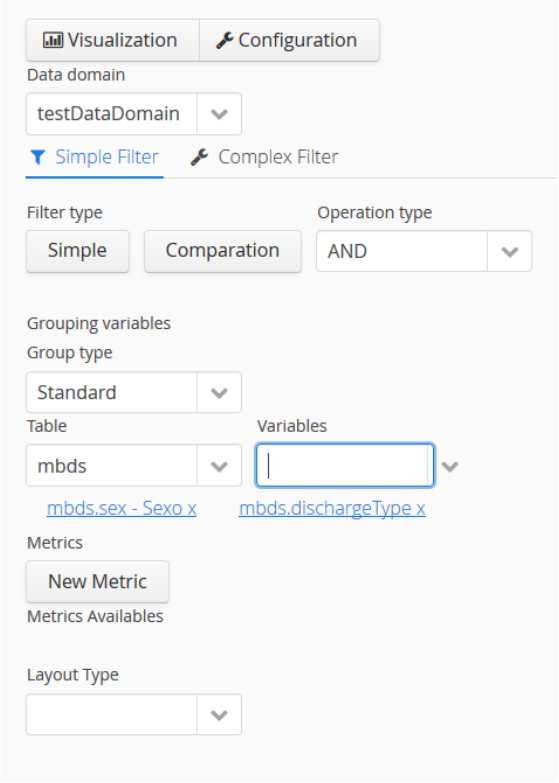

Imagen 29: Paso 6-1 Creación de la métrica

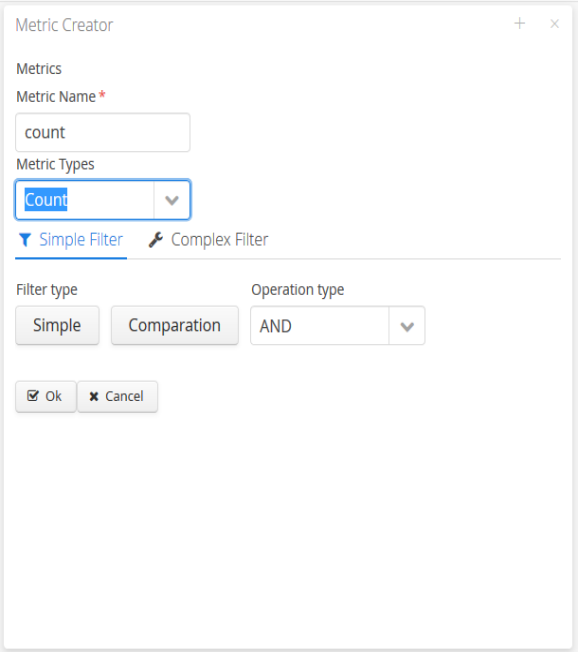

Imagen 30: Paso 6-2 Creación de la métrica

Paso 7

Este paso tiene 2 vertientes, si escogemos el tipo de layout que elijamos, si elegimos el tipo Drag & Drop hay un paso antes de crear la gráfica.

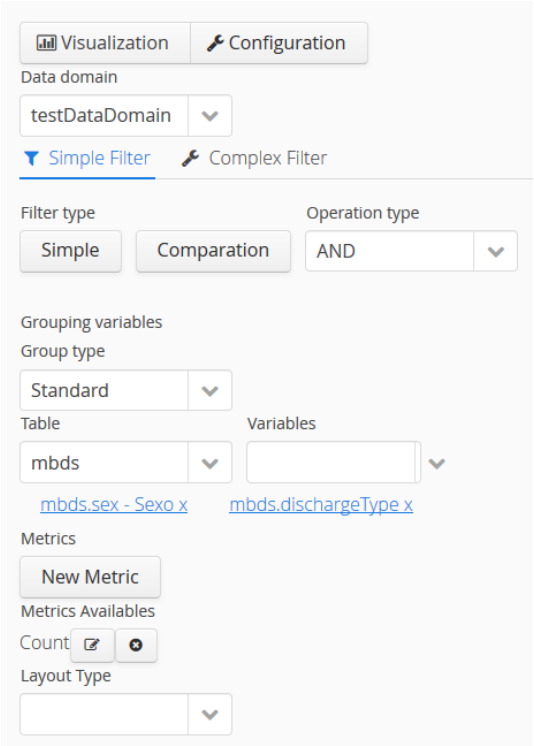

Imagen 31: Paso 7 Elección del layout de la gráfica

Paso 8

Si hemos escogido el layout de tipo *drag&drop* en el paso anterior habrá que elegir el tamaño de la gráfica.

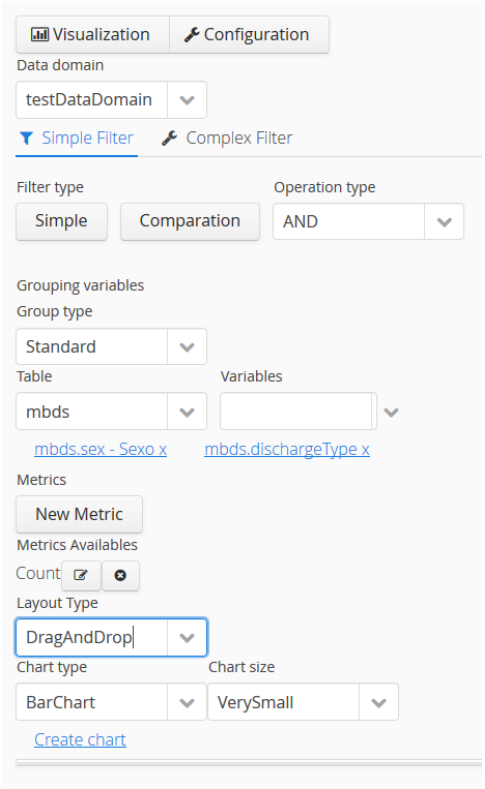

Imagen 32: Paso 8 Elección del tamaño de la gráfica

Paso 9

Por último sólo hay que elegir el tipo de gráfica y clicar en Create Chart para crear la gráfica.

El resultado de esos pasos sería el siguiente:

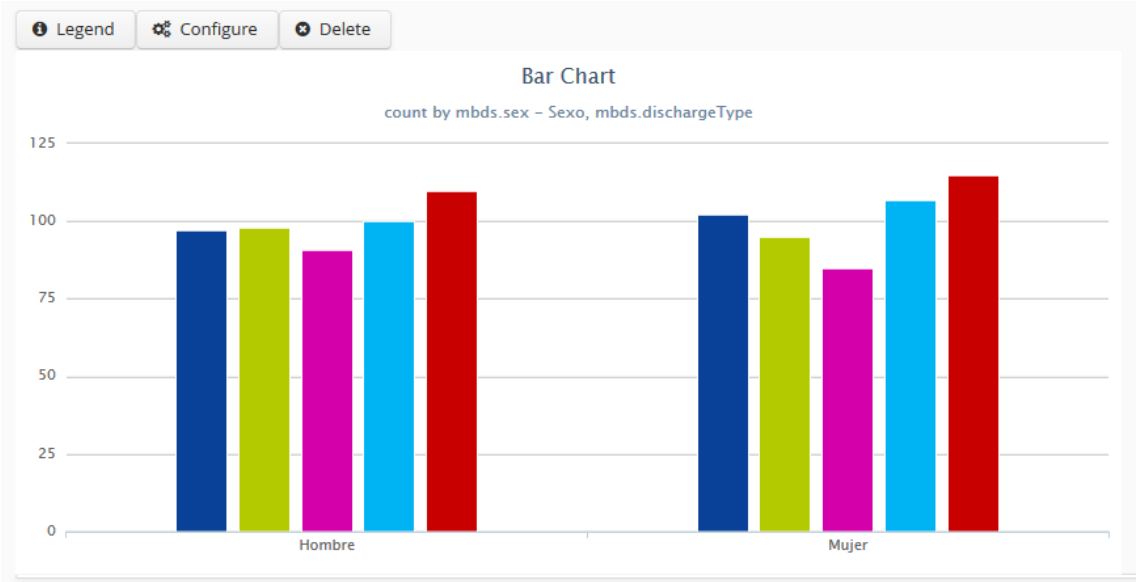

Imagen 33: Resultado Ejemplo

Por defecto las gráficas aparecen sin la leyenda, si la quisiéramos mostrar, habría que hacer clic en el botón "Legend" que aparece arriba a la izquierda de la gráfica, este menú es individual a cada gráfica y permite elegir si se muestra o no la leyenda, realizar tareas de configuración, o eliminarla.

Así sería la gráfica con la leyenda visible:

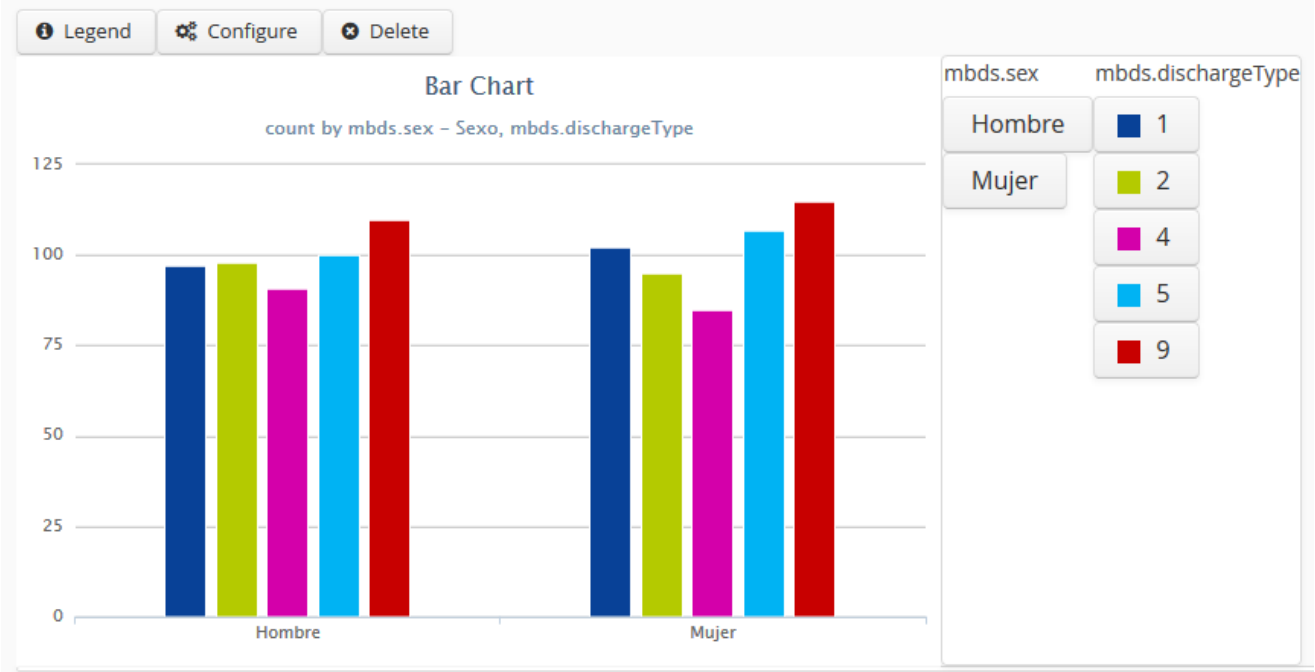

Imagen 34: Gráfica con leyenda

Para ocultar algún valor de alguna variable basta con pulsar en el botón correspondiente de la leyenda. Si queremos ocultar, por ejemplo, a los hombres y el tipo de alta 1, la leyenda quedaría de este modo:

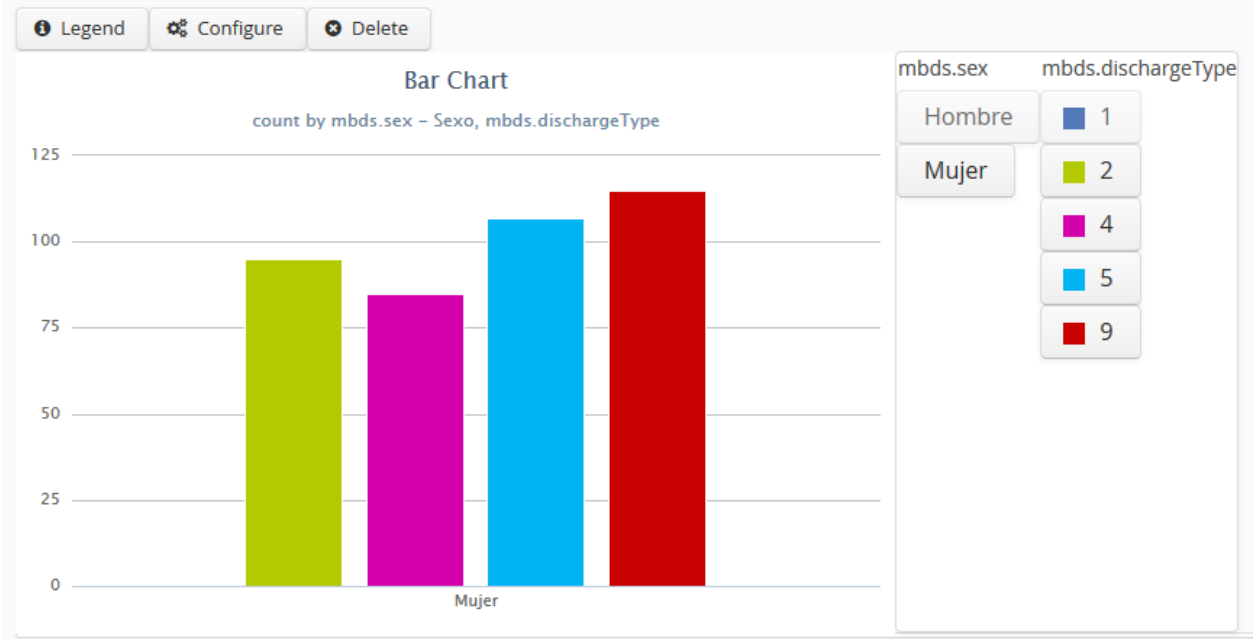

Imagen 35: Gráfica con valores ocultos

Se puede apreciar que los botones de los valores ocultos quedan con un tono grisáceo para indicar que los valores están ocultos.

Ahora vamos a proceder a añadir otra gráfica, en este caso de tipo barras apiladas y con una variable común a ambas gráficas. Para ello repetimos los pasos 3, 4, 5 y 9 para crear otra gráfica.

Quedando así:

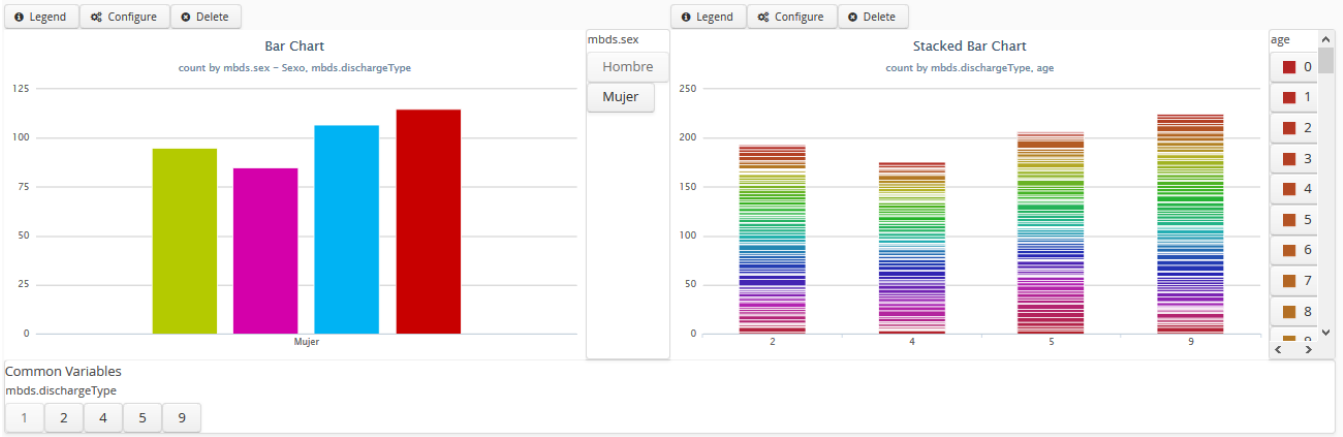

Imagen 36: Dos gráficas con leyenda común

Como se puede comprobar, al haber una variable común en ambas gráficas, en este caso dischargeType, aparece una leyenda común en la que se encuentra dicha variable

y desaparece de las leyendas individuales de cada gráfica. Las variables que no son comunes permanecen en las leyendas individuales de cada gráfica.
Se han mantenido ocultos los valores que habíamos ocultado después de crear la primera gráfica.
Si eliminamos la segunda gráfica con el botón “Delete”, la primera gráfica se quedaría igual que antes de añadir la segunda, la variable dischargeType volvería a su leyenda.

En el caso de haber elegido el tipo de layout *drag&drop* el comportamiento de la leyenda sería el mismo, la única diferencia es que si clicamos en una gráfica y la arrastramos delante o detrás de otra, cambia su posición, pero este comportamiento no se puede mostrar sobre el papel.

Configuración de gráficas

Ahora se va a explicar cómo configurar una gráfica.

Para configurar la gráfica hay que pulsar en el botón configure que se encentra encima de la gráfica como podemos observar en alguna de las imágenes anteriores. Una vez clicado aparecerá un menú con los valores disponibles de configuración.

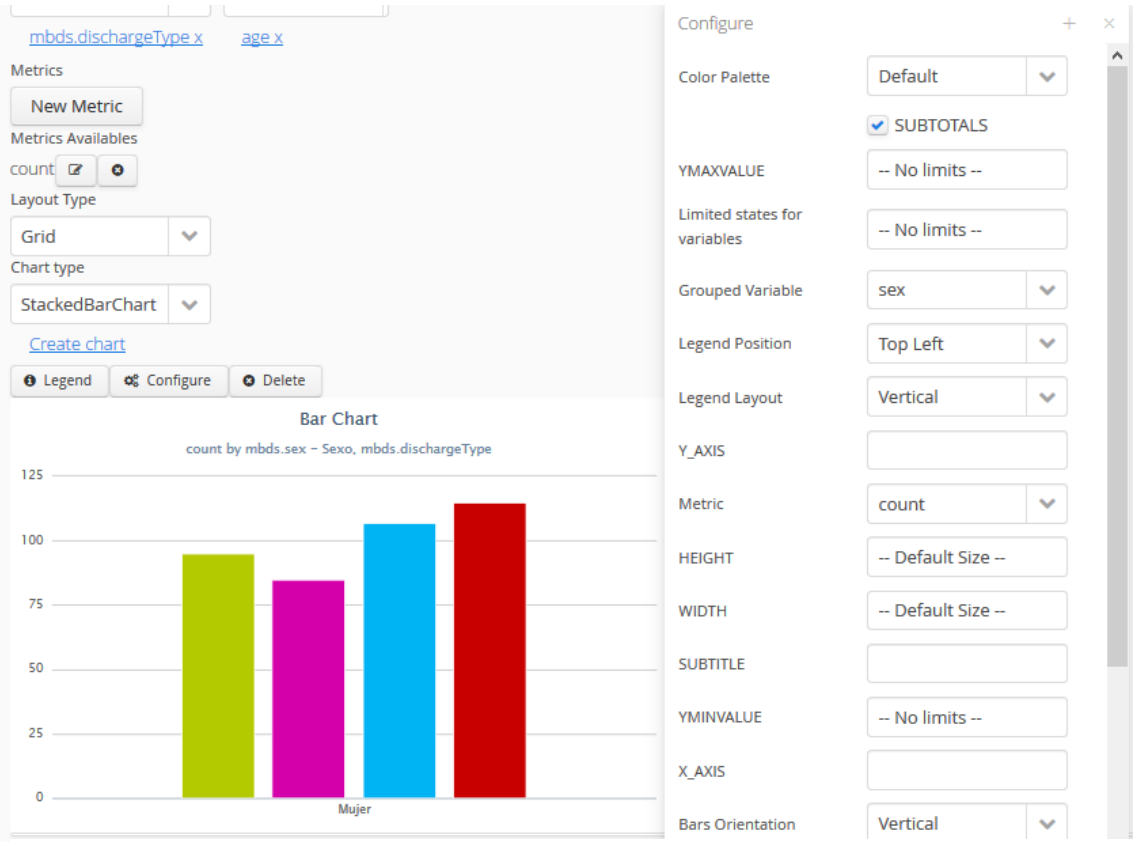

Imagen 37: Menú de configuración de la gráfica

Probaremos a cambiar los colores de la gráfica por la paleta Yellow-Red, cambiar la variable de agrupamiento por la otra de la gráfica, cambiar la orientación de las barras y quitar las líneas delimitadoras de unidades.
Tras estos cambios la gráfica quedaría tal que así:

Imagen 38: Gráfica personalizada

Agrupamiento de variables

En algunas ocasiones se quiere conocer también los totales en las distribuciones, por lo que existe la posibilidad de realizar sumarizaciones. Vamos a mostrar dos ejemplos, uno con ROLLUP y otro con CUBE para las variables que hemos utilizado para la primera gráfica. Para realizar gráficas con sumarizaciones los pasos a seguir son los mismos que arriba, a diferencia de que en el paso 5 en vez de elegir tipo de agrupamiento standard, se elige uno de las dos.

ROLLUP permite generar subtotales de la variable por la que se realiza la agrupación, en este caso el sexo. Los totales vienen indicados por la barra lila.

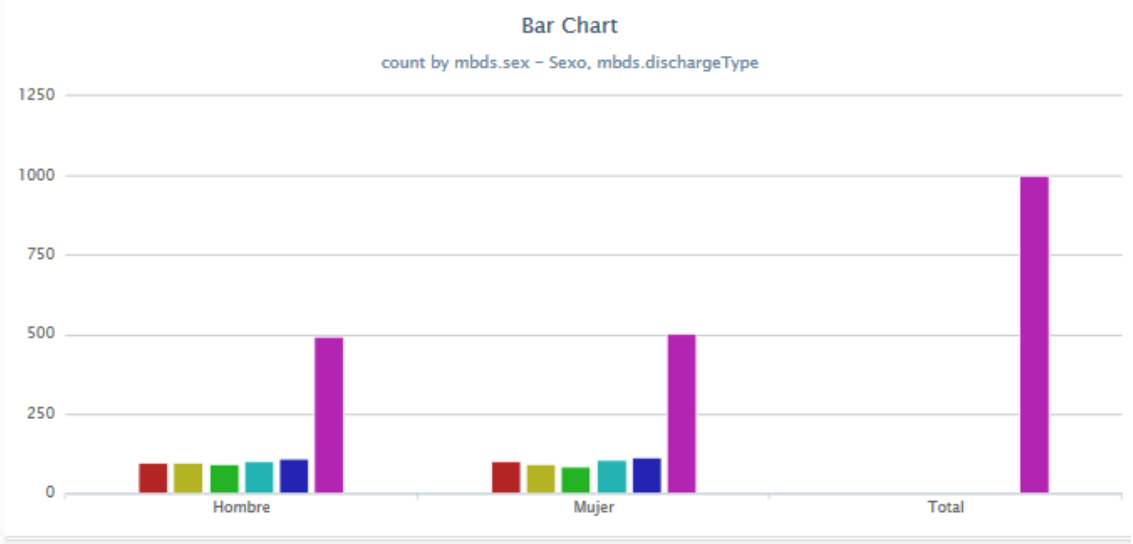

Imagen 39: Gráfica con ROLLUP

Por el contrario con CUBE permite generar subtotales para las diferentes combinaciones de la variable por la que se realiza la agrupación, en este caso el sexo.
Por lo tanto se generarán subtotales para cada combinación de los valores de la variable sex con los valores de la variable dischargeType. Los totales vienen indicados por la barra lila. En el agrupamiento Total aparecen los subtotales por cada valor de dischargeType.

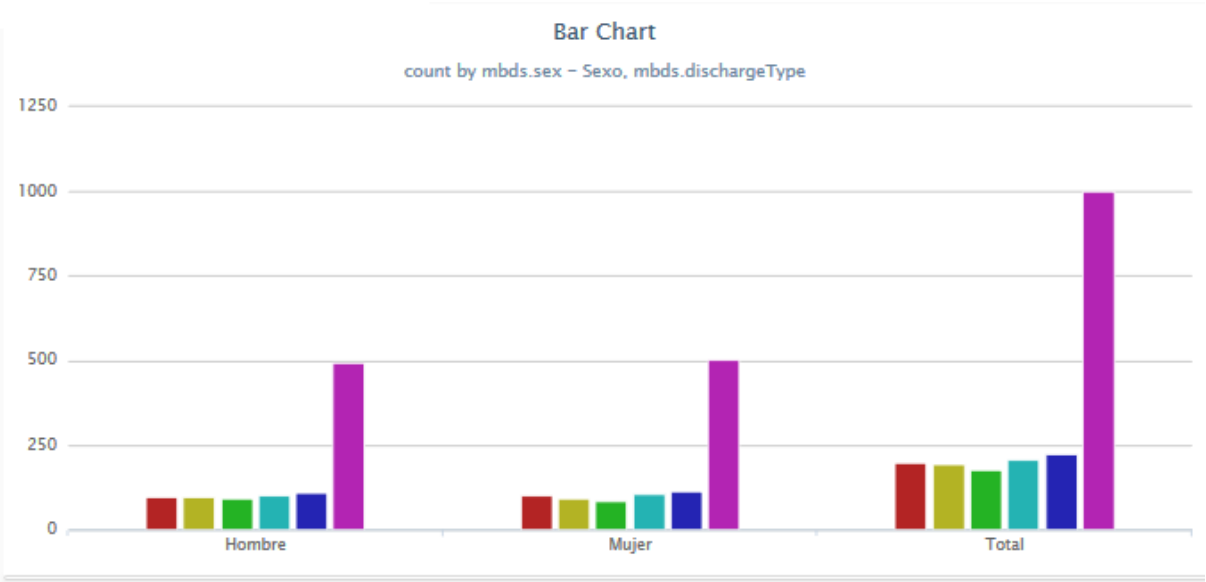

Imagen 40: Gráfica con CUBE

7. Conclusiones y trabajos futuros

7.1. Conclusiones

Este proyecto tenía como objetivo el desarrollo de una aplicación, basada en una interfaz web, para el análisis de datos mediante distintas gráficas. Tras haber realizado el diseño de la aplicación y su planificación y de realizar la implementación y pruebas de cada una de las iteraciones, he llegado a la satisfactoria realización del proyecto.

Tras el desarrollo del proyecto los objetivos alcanzados han sido:

- El diseño e implementación de una interfaz para poder crear gráficas y así analizar datos de una forma sencilla.
- El diseño e implementación de sumarizaciones de los datos para la agrupación de los mismos.
- La posibilidad de tener varias gráficas a la vez en pantalla para una comparación de datos o ver varios datos a la vez.
- El diseño e implementación de la personalización de las gráficas para ponerlas al gusto del usuario.
- El diseño e implementación de una leyenda para la identificación y el ocultamiento de variables de las gráficas.
- El diseño e implementación de varios modos de presentar las gráficas, en concreto una basada en un grid con varias gráficas por fila y otra en al que se pueden reordenar las gráficas a nuestro gusto mediante la técnica de “drag & drop” arrastrar y soltar la gráfica hasta la posición deseada.

Me gustaría destacar la formación que me ha proporcionado el desarrollo del proyecto ya que me ha dado una idea de la envergadura de un proyecto, comparado con las prácticas que realizamos durante la universidad, pasando por todos los pasos de un desarrollo software y viendo lo importante que son los requisitos en un desarrollo, aunque en un principio no les demos mucha importancia.

7.2. Trabajos futuros

Aunque la aplicación es plenamente funcional, siempre hay cosas que se podrían mejorar o cosas que se podrían añadir para hacerla un poco mejor.
A continuación expongo algunas de ellas.

7.2.1 Traducción de la interfaz

La interfaz de la aplicación viene en inglés, esto es debido a que es uno de los idiomas más extendidos en el mundo y a que es el idioma universal en la informática.

Aun así, mucha gente no sabe inglés o quiere que la aplicación esté en otro idioma debido a requisitos legales o de la propia empresa. Esto sería una tarea fácil debido a que tanto los textos que se muestran en la aplicación como los textos de los botones se obtienen de unos archivos de configuración en los que viene la traducción del mensaje dependiendo del idioma.

Se podría implementar mediante un selector de idiomas en la aplicación o mediante el idioma local de la máquina desde al que se usa la aplicación. Luego habría que rellenar ese archivo con las traducciones en los diferentes idiomas.

7.2.2. Exportación de las gráficas

En este momento la única forma de poder guardar las gráficas para una posterior consulta es la de realizar una captura de la pantalla para guardar lo que se muestra.

Uno de los posteriores desarrollos sería añadir la posibilidad de exportar las gráficas, que implementa Vaadin en su API, por lo que se facilita en gran medida la exportación de gráficas.

7.2.3. Añadir más tipos de gráficas

Actualmente la aplicación cuenta con 7 tipos de gráficas: barras, burbujas, líneas, tarta normal y múltiple, barras apiladas y temporal. Pero en Vaadin aún quedan algunos tipos más como las gráficas de área, las combinadas, los diagramas de caja o gráficas polares, por lo que en un futuro se podrían añadir al catálogo de tipos de gráficas disponibles.

8. Agradecimientos

Me gustaría dar las gracias a mis compañeros en Predictia.

- A Fernando Magaldi ya que fue el que desarrolló la interfaz base sobre la que se encuentra la que he desarrollado en mi proyecto, así como la interfaz para la configuración de las variables.
- A Daniel San Martín por dirigirme este proyecto y ayudarme en las dudas sobre el desarrollo de la aplicación que me fueron surgiendo.
- A Max Tuni que me ayudaba con los pequeños problemas que iba encontrando y me descubrió el mundo de los predicados, que ayudan mucho a la hora de filtrar colecciones.

Tampoco olvidarme de Carlos Blanco, mi director de la universidad por ayudarme con el desarrollo de la memoria que es una parte muy importante del proyecto, ya que sin ella los demás no pueden llegar a comprender el alcance y la composición del proyecto.

Bibliografía y referencias

Bibliografía

Grönroos, Marko. *Book of Vaadin.*
Sommerville, Ian. *Ingeniería del Software.* 9. s.l. : Addison-Wesley, 2012.
I.Neustadt, J.Arlow e. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design.* s.l. : Addison-Wesley, 2005.
D.Pilone. *UML 2.0 in a Nutshell.* s.l. : O'Reilly, 2005.
Cockburn, Alistair. *Writting Effective Use Cases.* s.l. : Addison-Wesley, 2000.

Referencias

1. proyectosagiles.org. [En línea] [Citado el: 8 de Diciembre de 2015.] http://proyectosagiles.org/desarrollo-iterativo-incremental/.
2. IBM Developer works. [En línea] [Citado el: 8 de Diciembre de 2015.] https://www.ibm.com/developerworks/ssa/java/newto/.
3. MySQL - Wikipedia. [En línea] [Citado el: 13 de Diciembre de 2015.] https://es.wikipedia.org/wiki/MySQL.
4. Álvarez, Cecilio. Genbeta Dev. [En línea] [Citado el: 13 de Diciembre de 2015.] http://www.genbetadev.com/javascript/highcharts-graficas-con-javascript.
5. Pesquera, Carlos. carlospesquera.com. [En línea] [Citado el: 8 de Diciembre de 2015.] http://carlospesquera.com/vaadin-aplicaciones-web-java-como-si-fuesen-de-escritorio/.
6. Subversion (software) - Wikipedia. [En línea] [Citado el: 13 de Diciembre de 2015.] https://es.wikipedia.org/wiki/Subversion_(software).
7. Apache Fundation Wikispace. [En línea] [Citado el: 13 de Diciembre de 2015.] https://apachefoundation.wikispaces.com/Apache+Tomcat.
8. calendamaia. Genbeta Dev. [En línea] [Citado el: 13 de Diciembre de 2015.] http://www.genbetadev.com/herramientas/eclipse-ide.