Recibido: 31/8/2021 / Aceptado: 11/10/2021
doi: https://doi.org/10.26439/interfases2021.n014.5394

EchartsTag para generación de *charts* usando Taglib

David Mamani-Pari
davidmp@upeu.edu.pe / ORCID: 0000-0002-5836-2018
Universidad Peruana Unión, Juliaca, Perú

Richart Smith Escobedo-Quispe
rescobedoq@unsa.edu.pe / ORCID: 0000-0003-3618-6858
Universidad Nacional San Agustín de Arequipa, Arequipa, Perú

Resumen

Existen muchas herramientas para desarrollar aplicaciones en Java, pero existen pocas alternativas para *framework* o bibliotecas especializadas para mostrar gráficos (*charts*) en entornos web. Aunque la mayoría son para entornos de escritorio, las librerías PrimeFaces y ChartistJSF son una alternativa para la web, pero están orientadas a proyectos basados en Java Server Faces. En este sentido, muchos proyectos prefieren utilizar principalmente una biblioteca de gráficos integrada con JavaScript, lo que implica una curva de aprendizaje adicional y requiere más tiempo en el desarrollo de aplicaciones. Por lo tanto, es necesario tener una biblioteca o marco de visualización de gráficos único para aplicaciones Java en el entorno web. Por lo tanto, en este artículo se propone construir una herramienta que ayude a desarrollar visualizaciones gráficas en proyectos web Java utilizando los conceptos de TagLibs y Webjars. La herramienta propuesta se llama EchartsTag y fue construida siguiendo la metodología XP y validada con SonarQube y un grupo de profesionales expertos en desarrollo de proyectos Java. EchartsTag está disponible en GitHub, por lo que puede ser utilizado por cualquier desarrollador de la comunidad de Java u organizaciones que trabajen con tecnología Java. Finalmente, los resultados de rendimiento de EchartsTag se demuestran al compararlo con otras herramientas alternativas, logrando un tiempo promedio de 14,17 minutos en tiempo de desarrollo y ocupando el tercer lugar en tiempo promedio de visualización que es 3,5 m después de Chartjs y HighChats. Además, ocupa el primer lugar en la evaluación de otros criterios para el desarrollo web java con herramientas de visualización de gráficos.

PALABRAS CLAVE: *charts* / Java Server Faces / EchartsTag / TagLibs, Webjars / aplicaciones Java / JavaScript

ECHARTSTAG FOR CHART GENERATION USING TAGLIB

Abstract

There are many tools for developing applications in Java, but there are few alternatives for frameworks or specialized libraries for displaying charts in web environments. Although most of them are for desktop environments, the PrimeFaces and ChartistJSF libraries are an alternative for the web, but they are oriented to projects based on Java Server Faces. In this sense, many projects prefer to use a graphics library built-in JavaScript, which implies an additional learning curve and is more time-consuming in developing applications. Therefore, it is necessary to have a unique graphics display library or framework for Java applications in the web environment. Therefore, in this article, it is proposed to build a tool that helps develop graphical visualizations in Java web projects using the concepts of TagLibs and Webjars. The proposed tool is called EchartsTag, and it was built following the XP methodology and validated with the SonarQube product and a group of professional experts in Java project development. EchartsTag is available on GitHub, so it can be used by any developer in the Java community or organizations that work with Java technology. Finally, the performance results of EchartsTag appear when compared with other alternative tools, achieving an average time of 14.17 minutes in development time and occupying third place in average viewing time that is 3.5ms after Chartjs and HighChats. In addition, to occupy the first place in evaluating other criteria for java web development with charts tools.

KEYWORDS: charts / Java Server Faces / EchartsTag / TagLibs / Webjars / Java application / JavaScript

1. INTRODUCCIÓN

Según Stack-overflow (2020), el 55,2 % son desarrolladores *Back-end* y el 37,1 % son desarrolladores *Front-end*, mientras que el 54,9 % son desarrolladores *full-stack*; esto evidencia mayor cantidad de desarrolladores en aplicaciones web, seguido por los desarrolladores en aplicaciones empresariales de entorno escritorio con el 23,9 %, y en cuanto a desarrollo móvil, cuenta con el 19,2 % de desarrolladores. Por consiguiente, se puede deducir que el desarrollo web tiene una mayor cuota de mercado. Este incremento de desarrolladores se viene dando desde años atrás, como se menciona en el trabajo de Yadav *et al*. (2018), motivado por la revolución digital. Por otro lado, Molina-Rios *et al*. (2020) refieren que el desarrollo de aplicaciones web está en su apogeo debido al avance de las tecnologías y la constante dependencia del internet. Así mismo Marashdih *et al*. (2019) refieren que el uso de aplicaciones web ha aumentado y se ha convertido en un estándar para representar datos y realizar lanzamiento de servicios.

Sulova (2019) considera que el análisis de datos es cada vez más importante para las empresas, además refiere que la mayoría de las aplicaciones de *software* están basadas en la web. Por lo que toma un mayor interés el desarrollo de aplicaciones web en las distintas organizaciones. Según las estadísticas de Stackoverflow (2020), Java es el tercer lenguaje de programación más usado para entornos web, después de JavaScript y Python. Sin embargo, hay pocas alternativas de marcos de trabajo para *charts* en Java web, a pesar de que existen muchos *frameworks*; sin embargo, tienen otros propósitos y, en cuanto a *charts* se tiene a PrimeFaces y ChartistJSF que están orientadas a tecnologías de Java Server Faces que usa la estructura XHTML para el lado del usuario; en razón de ello, se hizo una revisión de literatura respecto a *charts* para Java obteniendo los resultados que se muestran en la tabla 1, donde la mayoría de bibliotecas son para entornos de escritorio; sin embargo, para entornos web, según Burtini *et al*. (2013), generalmente se usa una biblioteca o *framework* escrito en JavaScript lo que conlleva un tiempo adicional de aprendizaje; en consecuencia, se requiere una herramienta especializa en Java web para el desarrollo de *charts* interactivos para la visualización de la información que ayude a la toma de decisiones. Aunque en el año 2007 se hizo un trabajo para solucionar este problema, sin embargo, no se encuentra disponible la biblioteca en la web (Ostruszka *et al*., 2007). Por lo que aún se requiere una herramienta que pueda ser una alternativa para el desarrollo de *charts* en el lenguaje de programación Java web.

Por otro lado, Hua *et al.* (2018) mencionan que los humanos dependen cada vez más de la web basada en la visualización de datos para tomar decisiones en diversos aspectos como negocios, finanzas, educación, gobierno, industria e incluso con fines personales. Así mismo, Shen *et al*. (2019) refieren que antiguamente las páginas web eran estáticas, pero con la aparición de la web 2.0 se masificaron las visualizaciones de información con contenidos dinámicos; referenciando así las presentaciones más comunes entre una a

tres dimensiones de datos, los cuales son: Line Chart, Bar Chart, Pie Chart, Scatter Chart, Area Chart. De la misma forma, Sorapure (2019) manifiesta que las visualizaciones de datos están cada vez más dirigidas a usuarios cotidianos con una variedad de intereses y objetivos, y resalta, además, cuatro tipos de visualizaciones como son las imágenes, texto, datos e interacciones. Considerando, que las visualizaciones de información en la web toman un papel importante, se propuso el desarrollo de una biblioteca o marco de trabajo para facilitar el desarrollo de visualizaciones en la plataforma Java web, puesto que se dispone de pocas alternativas para tal propósito.

2. REVISIÓN DE LITERATURA

2.1 Herramientas para la visualización de charts en Java web

Según, Gilbert (2019), JFreeChart es una biblioteca libre u *open source* para Java que facilita el desarrollo de gráficos de calidad y está orientada a aplicaciones tanto para el lado del servidor como del cliente. Además, es compatible con los componentes de Swing y JavaFX y muestra los gráficos en archivos de imagen (PNG y JPEG) y formatos de archivos vectoriales como (PDF, EPS y SVG).

XChart es una biblioteca liviana, que se centra en la simplicidad y la facilidad de uso, ya que requiere pocas líneas de código para mostrar un gráfico y contiene una gran cantidad de colecciones de tipos de gráficos y son compatibles con las tecnologías Swing de Java, Java EE, y Java SE, pero no está orientada a entorno web (Knowm Inc., 2019).

PrimeFace es un *framework* UI, es decir, es un marco de trabajo de interfaz de usuario que tiene varios componentes; entre ellos está Charts y Chartjs que contiene varios gráficos principales; cabe resaltar que es un librería única que no tiene dependencias adicionales y es muy simple de usar y está orientada a entornos web; sin embargo su compatibilidad es con los proyectos de tipo Java Server Faces (JSF) y está patrocinada por PrimeTek Informatics; como uno de los proyectos de código abierto, bajo la licencia gratuita de Apache (PrimeTek, 2019).

ChartistJSF está relacionado con gráficos receptivos, altamente personalizables para Java Server Faces. Se basa en Chartist.js para generar los gráficos en SVG puro, además es una Api de gráficos simple. Donde hace una diferencia con PrimeFaces que es un poco más tedioso ya que está basado en jqPlot para la generación de gráficos que limita la manipulación en los elementos del DOM (Alimam, 2017).

Redko (2014) menciona que JavaFX Chart es un componente de interfaz de usuario desarrollado para uso general en aplicaciones de sistemas de información, y los gráficos disponibles en JavaFX Chart como parte del paquete de SDK de JavaFX contiene los siguiente gráficos: Pie Chart, Line Chart, Area Chart, Bubble Chart, Scatter Chart y Bar

Chart; los mismo que tienen sus ejemplos correspondientes; sin embargo, están orientados hacia aplicaciones de entorno escritorio como Swing y JavaFX.

Finalmente, las herramientas anteriores citadas son especializadas para Java; algunos para entornos web, otros para entornos de escritorio y algunos con soporte para ambos entornos, como es el caso de JFreeChart lanzado el año 2000, que es una de las más usadas en Java y dicha afirmación se puede corroborar mediante las estadísticas de tendencias de Google Trends como se observa en la figura 1. Sin embargo, JFreeChart para entornos web se generan gráficos en formato de imagen, lo que hace que no sean gráficos interactivos; como sí es posible hacerlos con herramientas para *charts* desarrolladas en JavaScript. Por lo tanto, esta es una razón para crear una herramienta que permita lograr gráficos interactivos en Java web, sin necesidad de aprender a programar en JavaScript.

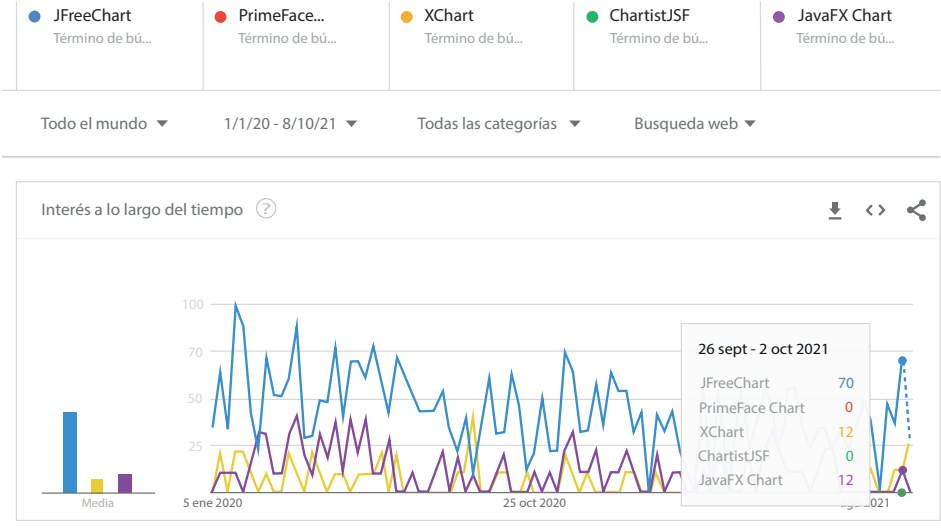

Figura 1. Resultados de tendencias de búsqueda sobre herramientas Chart para Java.

Fuente: captura de Google Trends.

2.2 Desarrollo de Framework o librería

El proceso de desarrollo de un *framework* no es igual al desarrollo de aplicaciones tradicionales y cuenta con ciertos procesos o etapas principales de desarrollo de *framework* como es: el análisis del dominio, diseño de *framework* y la instanciación del *framework* (Rodriguez, 2007). Véase la figura 2.

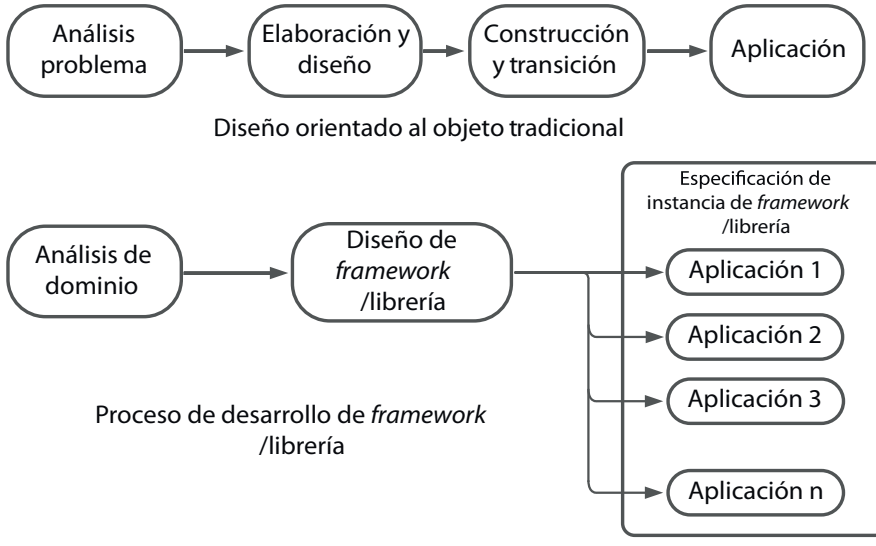

Figura 2. Proceso de desarrollo del *framework*/librería

Fuente: Adaptado de Rodríguez (2007).

2.3 *Charts*

Son gráficas para representar alguna información como cuadros estadísticos que juegan un papel primordial en diferentes áreas de la vida, como la información, la educación, la comunicación o la investigación (Martínez *et al.*, 2019). Para ello existen muchas herramientas en la programación web, que generalmente están escritas en JavaScript como es el caso de ECharts que es un *framework* de visualización gráfica en el que también se hizo un estudio de comparación con las otras herramientas similares como HighCharts, C3.js, Chart.js que evaluó algunas características como el soporte de canvas, personalización de interacciones, entre otros, saliendo como ganador ECharts (Li *et al.*, 2018).

2.4 Taglib

Son herramientas para crear nuevas etiquetas en Java web; mediante la invocación de la clase TagSupport que es la clase base para la creación de nuevos controladores de etiquetas o Taglibs (Oracle, 2019). La clase TagSupport implementa las interfaces de Tag e InterationTag y agrega algunos métodos de obtención para las propiedades de Tag. La creación de nuevas etiquetas ayuda en la portabilidad de los componentes y también la simplicidad para los desarrolladores, contribuyendo de esa manera al lenguaje expresivo (EL) (Oracle, 2021). Los controladores de etiquetas deben ser uno de los dos tipos: controladores de etiquetas clásicas que proporciona el TagSupport o BodyTagSupport en caso de que la etiqueta utilice un cuerpo; estas se definen en un archivo descriptor de

librerías TLD, que describe la sintaxis de cada etiqueta y los relaciona con las clases de Java que son los controladores para poder ejecutar su funcionalidad.

3. METODOLOGÍA

Según Wells (2013), la programación extrema (XP) es uno de varios procesos ágiles y el más popular. Así mismo, dicha metodología demuestra tener mucho éxito en las distintas organizaciones, grandes, pequeñas y medianas, debido a que busca la satisfacción del cliente en lugar de entregar todo lo que se desea en una fecha lejana. Además, esta metodología busca entregar el producto a medida que el cliente lo necesita y busca responder a los cambios en los requisitos del cliente, incluso al final del ciclo de vida. La programación extrema busca el trabajo en equipo y mejora un proyecto de *software* en cinco formas: comunicación, simplicidad, retroalimentación, respeto y coraje. Por otro lado, los programadores mantienen una comunicación constante con el cliente y sus compañeros de equipo, mantienen un diseño simple y limpio y, también, reciben comentarios al probar la aplicación y procuran entregar el producto lo antes posible al cliente para recibir retroalimentación. Según Goto *et al*. (2014), la programación extrema contempla las siguientes etapas: planificación, diseño, codificación y pruebas. Dichas etapas fueron consideradas en la construcción de EcharsTag que es la herramienta que se propone en el presente artículo. Por otro lado, es necesario aclarar que se eligió a XP como metodología de desarrollo, dado que su forma de trabajo es por fases y es la que mejor se ajustaba a la naturaleza del proyecto.

3.1 Planificación para el desarrollo de EchartsTag

Conforme a la fase de planificación de la metodología XP se identificaron las historias de usuarios, producto de la revisión de literatura con respecto a *charts* más usados y frecuentes en las distintas herramientas de visualización de información o visualización web para Java, como también se puede corroborar en las tendencias de búsqueda de Google Trends que se observa en la figura 1. Así, se pueden identificar los principales tipos de gráficos como: Bar Chart, Line Chart, Area Chart, Pie Chart, Scatter Chart y Radar Chart. Asimismo, hay ciertas diferencias en cada una de las herramientas como se puede apreciar en la tabla 1, por lo que se optó por agregar Funnel Chart y Boxplot Chart como elementos diferenciadores. Además de algunos *charts* en 3D como Bar 3D Chart, Line 3D Chart y Scatter 3D Chart.

Tabla 1
Herramientas Para Charts en Java

	JavaFX	JFreeChart	PrimeFace Chart	XChart	ChartistJSF
Pie chart	x	X	X	x	x
Line chart	x	X	X	x	x
Area chart	x	X	X	x	x
Bubble chart	x		X	x	
Scatter chart	x		X	x	
Bar chart	x	X	X	x	x
Radar chart			X	x	
Gantt chart		X			
Gauge chart			X		
Stack chart				x	
3D	NO	Bar chart	NO	NO	NO
Entorno	Desktop	Desktop	Web / JSF	Desktop	Web / JSF

Elaboración propia

Cada uno de los gráficos formó parte de la historia de usuarios, y se contó con once iteraciones, de acuerdo a la metodología XP, antes del lanzamiento del producto final. Sin embargo, previamente se hizo la especificación de historias de usuario que se muestra un ejemplo en la tabla 2; así mismo, se definió las pruebas de aceptación de los once tipos de gráficos, como se observa en la tabla 3; además se contempló el control de errores que estas pueden darse por el ingreso inadecuado de datos.

Tabla 2
Especificación de historia de usuario

Historia de usuario	
Número: 1	Usuario: desarrollador de Java web
Nombre de la historia: Visualizador gráfico de tipo Bar Chart para mostrar la información procesada	
Prioridad en negocio (alta, media, baja): Alta	Riesgo en desarrollo (alto, medio, bajo): Bajo
Puntos estimados: 2	Iteración asignada: 1
Programador responsable: investigador	
Descripción: Debe permitir visualizar el gráfico en forma de barras o histogramas, mostrar las etiquetas por colores y nombre según los datos mostrados en barra o histograma; mostrar los valores en cada barra o histograma, contemplar un deslizador para ambos ejes a fin de tener claridad en visualización; así mismo debe permitir la descarga del gráfico en formato de Imagen.	

(*continúa*)

(*continuación*)

Observaciones: En la parte de desarrollo se debe contemplar la opción de configurar el idioma con la que se desea que se visualice, como también el tipo de estilo de colores de la gráfica a visualizar.

Elaboración propia

Tabla 3
Prueba de aceptación

Prueba de aceptación	
Código: 1	N. ° historia de usuario: 1
Historia de usuario: visualizador de gráfico en Tipo Bar Chart para visualización de información procesada.	
Condiciones de ejecución: En el desarrollo se debe colocar de forma obligatoria los datos en los siguientes elementos de la etiqueta: echartBar: dataLabel, dataValuesEjeBase, dataValues, idCharts. echartBarHistogram: idCharts, chartTitle, legendDataName, ejeDataX Pero primero debió referenciar el Taglib: <%@taglib uri="http://www.syscenterlife.com/echarts" prefix="echar" %> Y luego la siguiente etiqueta: <echar:echartHeaderScript/>	
Entrada / pasos de ejecución: Elegir el tipo de gráfica mediante un clic. Visualizar la gráfica de acuerdo con los parámetros o datos ingresados en los elementos de la etiqueta.	
Resultado esperado: Mostrar la gráfica de acuerdo al tipo de gráfica seleccionada.	
Evaluación de la prueba: La prueba se concluyó satisfactoriamente.	

Elaboración propia

3.2 Diseño de EchartsTag

En la etapa de diseño se definió la estructura general del proyecto contemplando los recursos, código fuente y las pruebas unitarias, como se aprecia en la figura 3, y en seguida se definió la responsabilidad de colaboración de clases como se observa en la figura 4.

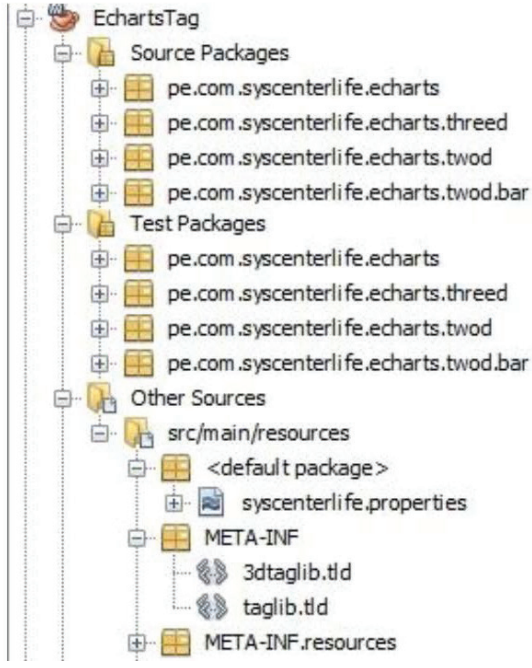

Figura 3. Estructura de Proyecto EchartsTag

Elaboración propia

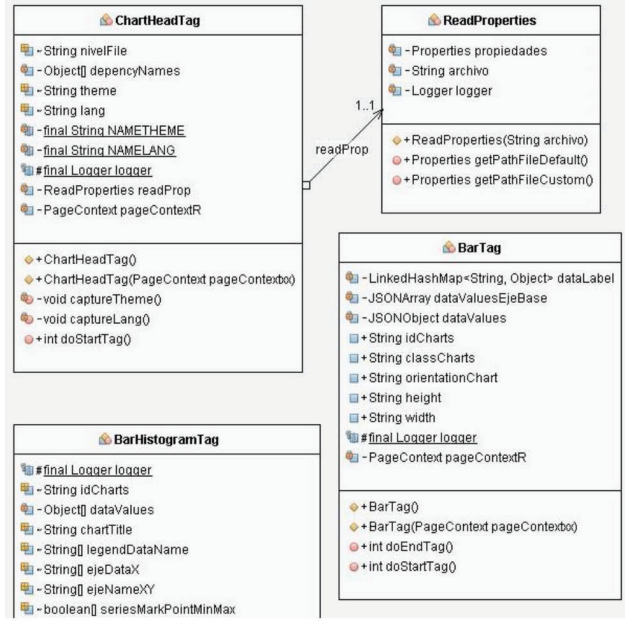

Figura 4. Diagrama de la clase de colaboración de responsabilidad (CRC)

Elaboración propia

3.3 Codificación de EchartsTag

En esta fase se procedió a programar de acuerdo con la especificación de historias de usuarios en relación con las iteraciones programadas y en función a la estructura definida en el paso anterior; es así que se inicia con la definición de las dependencias en el archivo pom.xml, que es el archivo principal de configuración y manejo de dependencias en proyectos de tipo maven en Java. Usando las siguientes librerías o dependencias que se muestran en la tabla 4, además de precisar que no fue necesario el uso de muchos patrones de diseño por la simplicidad; a excepción del patrón de Singleton, se ha usado en el desarrollo, más sí se han contemplado muchas buenas prácticas que fueron evaluadas con la herramienta de SonarQube, que es un *software* que mide la calidad interna del código.

Tabla 4
Dependencias utilizadas para la construcción de EchartsTag

Nombre de dependencias	Versión	Funcionalidad	Tecnología
Junit	4.13.1	Para realizar pruebas unitarias del código	Java
JSON	20180130	Para intercambio de datos en formato Json	Java
lombok	1.18.4	Para implementar las funciones *setter* y *getter* mediante anotaciones para los atributos de los *charts*	Java
mockito	3.2.4	Para realizar pruebas en Taglibs	Java
Jacoco	0.8.6	Para reportes de pruebas unitarias	Java
sonar-scanner-maven	3.6.0.139	Para desplegar el proyecto en SonarQube	Java
xinghuangxu.sonarqube	5.1		Java
maven-plugins	3	Para compilado de proyectos maven	Java
javaee-web-api	7	Para referencias a Servlets dentro de Taglibs	Java
Echarts	4.6.0	Para generar gráficos *charts*	JavaScript

Elaboración propia

Para la construcción del EchartsTag se estableció el ambiente de desarrollo correspondiente; para este caso, fue la configuración de Maven que es un gestor de dependencias. El IDE de desarrollo que se utilizó fue Apache NetBeans, y el cliente de Git para el manejo de versiones. Se procedió así con la creación de las clases correspondientes de acuerdo con la estructura definida en el paso anterior, considerando las iteraciones de las historias de usuario y tomando en consideración los criterios de buenas prácticas de codificación; estos últimos fueron evaluados posteriormente con la herramienta de SonarQube. Una vez concluida la creación de las clases heredadas por BodyTagSupport y las funcionalidades correspondientes, se procedió a crear un archivo

con la extensión TLD que es un archivo descriptor de librerías para definir la sintaxis de cada una de las etiquetas relacionados con las clases definidas en Java que actúan como controladores para ejecutar una función en específica, como se puede verificar en el código fuente del proyecto, que está alojado en el repositorio de la plataforma Github, ya que es una de las herramientas más usadas por los desarrolladores.

Como parte de esta etapa también están las pruebas unitarias. Para ellas se recurrió a Junit y Mockito que son librerías para pruebas unitarias; sin embargo, el último permite el trabajo con objetos simulados, logrando así las pruebas unitarias de las clases que heredan de TagSupport. Asimismo, en la figura 5 se observan los resultados de las pruebas unitarias.

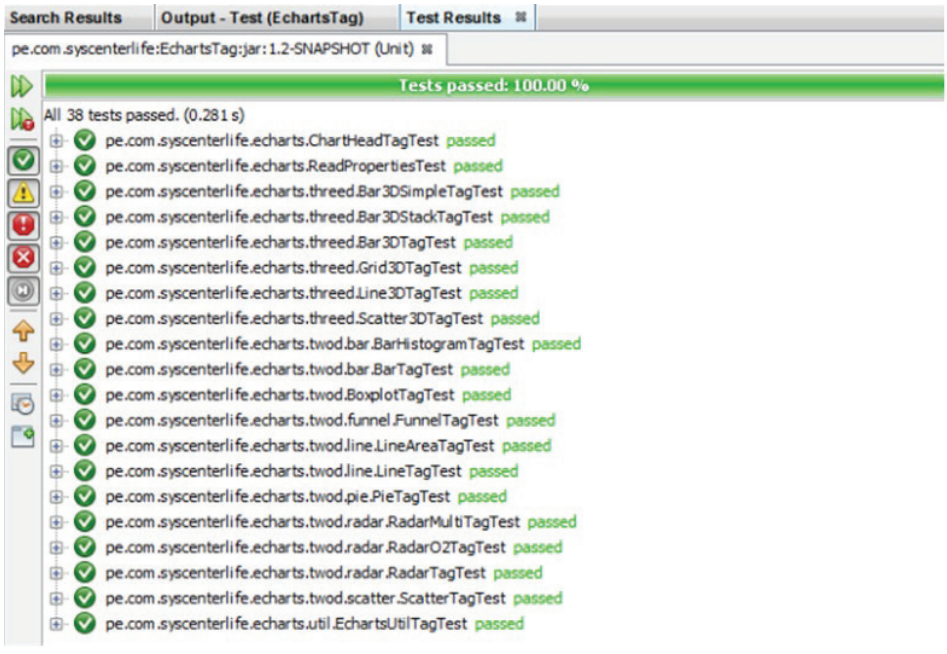

Figura 5. Resultados de la ejecución de pruebas Unitarias

Elaboración propia

3.4 Pruebas de EchartsTag

Siguiendo la metodología XP, finalmente se ejecutaron las pruebas de aceptación en relación con lo establecido en la etapa de planificación. Antes de ello se procedió a la creación de un proyecto demo donde se realizó la instancia a EchartsTag para crear *charts*; dicha instancia se hizo en un archivo denominado pom.xml del nuevo proyecto, como se observa en la figura 6.

```
22          <dependency>
23              <groupId>com.github.davidmp</groupId>
24              <artifactId>EchartsTag</artifactId>
25              <version>1.8</version>
26          </dependency>
```

Figura 6. Referencia de la dependencia de EchartsTag

Elaboración propia

En seguida se procedió a crear los ejemplos de *charts* usando la librería construida previa configuración del archivo syscenterlife.properties en el que se define el estilo de gráficos y el soporte de idioma. Posteriormente se referencio al Taglib de EchartsTag para la creación de gráficos, como se observa en la figura 7 y se puede ver la gráfica en la figura 8.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://www.syscenterlife.com/echarts" prefix="echar" %>
<!DOCTYPE html>
<html>
<head> <echar:echartHeaderScript /> </head>
  <body>
    <%
    Object[][] dataValuesX={
            {335,"Direct access"},
            {310,"Email Marketing"},
            {234,"Affiliate Advertising"},
            {135,"Video ad"},
            {1548,"Search engine"},};
    String chartTitle="Un usuario del sitio visita la fuente";
    %>
<echar:echartPie idCharts="main1" chartTitle="<%=chartTitle%>"
dataValues="<%=dataValuesX%>" />
</body>
</html>
```

Figura 7. Código de gráfico circular con EchartsTag

Elaboración propia

Figura 8. Resultado del gráfico circular con EchartsTag

Elaboración propia

Una vez desarrollados todos los ejemplos según tipos de gráficos para automatizar las pruebas de aceptación, en primer lugar, se procedió a usar Selenium, que es un entorno de pruebas basadas en la web que permite reproducir y grabar las pruebas; por lo que, se procedió a realizar la configuración correspondiente del WebDriver como se observa en la figura 9.

```
25
26      private WebDriver driver;
27
28      @BeforeMethod
29      public void setUP(){
30          DesiredCapabilities caps=new DesiredCapabilities();
31          System.setProperty("webdriver.chrome.driver", "D:\\TESTDRIVERS\\chromedriver.exe");
32          driver=new ChromeDriver();
33          driver.manage().window().maximize();
34          driver.navigate().to("http://localhost:8080/EchartsJsp/");
35              try {
36                  Thread.sleep(6000);
37              } catch (InterruptedException e) {
                    e.printStackTrace();
39              }
40      }
```

Figura 9. Configuración de WebDriver para la automatización de pruebas de aceptación

Elaboración propia

En seguida, se procedió a configurar las dependencias necesarias en el archivo pom.xml del proyecto EcharSelenium, los cuales son: el TestNG, Selenium y el HttpClient. Después de escribir el código de los *scripts* de pruebas de aceptación y al ser ejecutadas estas, se obtuvieron resultados satisfactorios, ya que el 100 % de las pruebas automatizadas pasaron correctamente, como se puede apreciar en la figura 10.

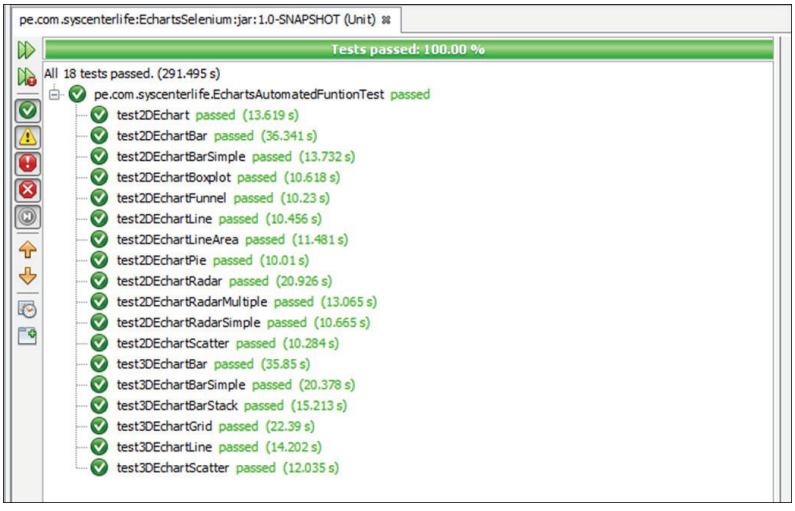

Figura 10. Resultados de Pruebas de Aceptación-Selenium

Elaboración propia

4. RESULTADOS

Finalizada la construcción de EchartsTag, se procedió a publicar el artefacto en el repositorio de Github (URL: https://github.com/davidmp/EchartsTag) inicialmente con la versión 1.2; actualmente está en la versión 1.8; esto permite enlazar con el gestor de paquetes Maven mediante JitPack como se observa en la figura 11. A fin de que cualquier desarrollador pueda hacer uso libremente de EchartsTag en sus aplicaciones web.

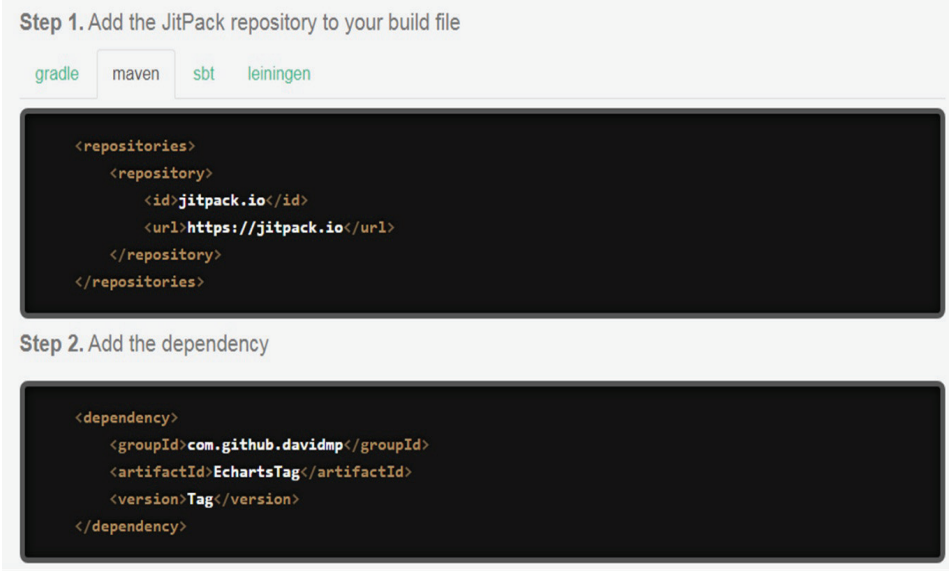

Figura 11. EchartsTag Publicado en JitPack

Elaboración propia

Finalmente se instanció EchartsTag en tres tipos de proyectos Java: JSP Plano, Servlets y SpringBoot, corroborando su compatibilidad con cualquier tipo de proyectos Java en entornos web, a excepción de proyectos de tipo Java Server Faces. Los ejemplos de los proyectos se pueden probar y verificar en el siguiente enlace del repositorio de Github. https://github.com/davidmp/ExampleFrameworkEcharts.

4.1 Creación de visualización web con EchartsTag

Los ejemplos se pueden encontrar en el enlace anterior, sin embargo aquí se hizo la demostración en uno de los tipos de proyectos, el de Java Server Page. Para ello previamente se tuvo que identificar un caso real, a fin de representar los datos en una visualización de *charts*, por lo que se eligió visualizar los diez países que tienen mayores certificaciones en las ISO a nivel mundial; en seguida se identificó el tipo de *chart* que mejor se adecua a la representación de datos, para este caso se eligió la etiqueta echartBarHistogram. Dicha representación, tanto a nivel de código y resultado se puede observar en las figuras 12, 13 y 14.

```
String chartTitle="Estadística de Certificaciones - 2018";
double[] dataValues1={295703,87794,47482,34335,31795,29562,26434,21848,21095,14123};
double[] dataValues2={136715,15118,8028,19131,7374,12198,11201,11201,4000,5777};
double[] dataValues3={11581,937,300,1283,1976,585,280,100,88,60};
double[] dataValues4={6443,332,147,120,525,138,928,100,90,80};

Object[] dataValues={dataValues1,dataValues2,dataValues3,dataValues4};
String[] legendDataName=new String[dataValues.length];
legendDataName[0]="ISO 9001";
legendDataName[1]="ISO 14001";
legendDataName[2]="ISO 22000";
legendDataName[3]="ISO 45001";
String[] ejeDataX={"China","Italia","Alemania","Japón","India","España","Reino Unido",
                   "Estados Unidos","Francia","República de corea"};
String[] ejeNameXY={"Eje X","Eje Y"};
boolean[] seriesMarkPointMinMax ={false,true,false,false};
boolean[] seriesMarkLineMedia ={false,false,true,false};
String[] seriesStackName ={"one","one","two","two"};
String echartsOriented="horizontal";/*vertical,horizontal*/
```

Figura 12. Fragmento de código en Java

Elaboración propia

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://www.syscenterlife.com/echarts" prefix="echar" %>
<!DOCTYPE html>
<html>
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
                <title>JSP Page</title>
                <echar:echartHeaderScript/>
        </head>
<body>
<% /*Here Java code*/ %>
<echar:echartBarHistogram  chartTitle="<%=chartTitle%>" dataValues="<%=dataValues%>" ejeDataX="
<%=ejeDataX%>"
                                idCharts="main" legendDataName="<%=legendDataName%>"/>
</body>
</html>
```

Figura 13. Fragmento en JSP

Elaboración propia

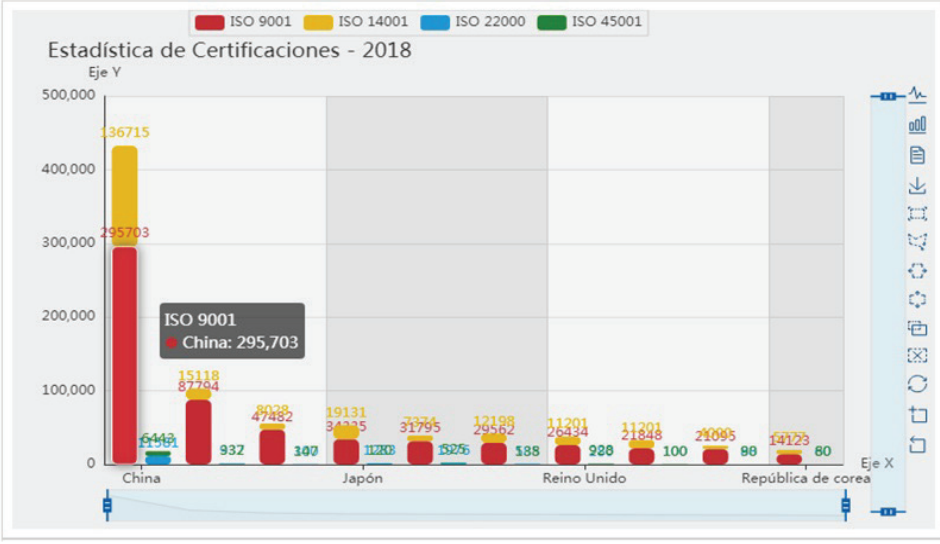

Figura 14. Resultado sobre países con mayores certificaciones

Elaboración propia

4.2 Validación

Para la validación de EchartsTag se consideró las dimensiones y atributos de la ISO 25. 010 que es uno de los estándares para evaluar la calidad de *software*. Por lo que tres de sus dimensiones fueron evaluadas con la herramienta de SonarQube y para el resto de las dimensiones se optó por la modalidad de Robiolo y Santos (2016) juicio de expertos, ya que es una de las técnicas más usadas en diversos campos como estimaciones de proyectos de *software*, negocios, salud, educación, productos de *software*, etc.

4.2.1 SonarQube Software para evaluar calidad de código

Una vez concluido todo el desarrollo según la metodología XP, se procedió a validar el código fuente, para ello se usó el *software* SonarQube, que es una herramienta especializada para evaluar calidad interna de código; se obtuvieron resultados favorables de un nivel A en cuanto a fiabilidad, seguridad y mantenibilidad, como se observa en la figura 15. Para lograr dicho resultado evidentemente se tuvo que evaluar y corregir las partes que generaban alguna inseguridad o problemas de mantenimiento, etc. como también ver la cobertura de código por las pruebas unitarias.

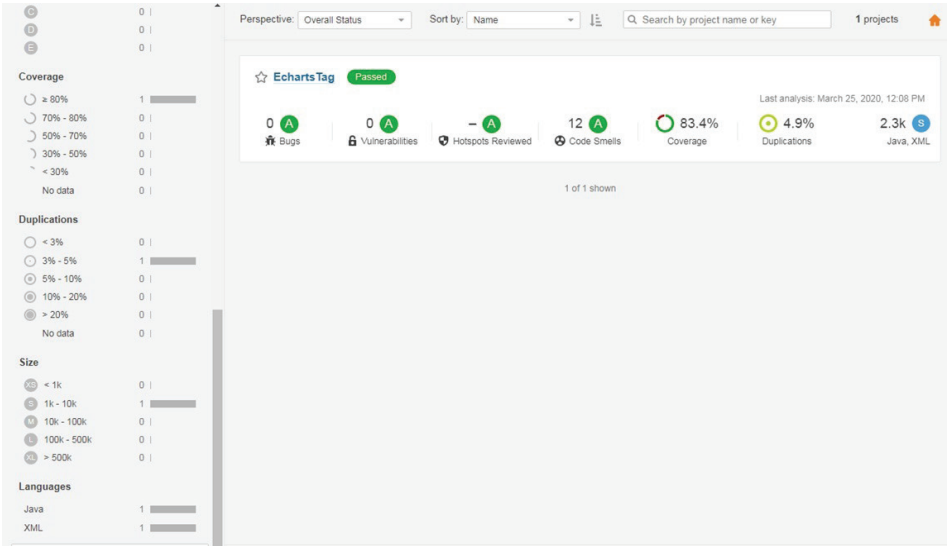

Figura 15. Resultado de la evaluación de atributos de calidad de EchartsTag

Elaboración propia

4.2.2 EchartsTag – Validación por juicio de expertos

Respecto a las dimensiones de adecuación funcional, eficiencia de desempeño, compatibilidad, usabilidad, portabilidad, productividad y satisfacción se evaluó en función a los atributos de la ISO 25 010 y en función a ello se formularon las preguntas (ver anexo A) tomando como base teórica el estándar en mención, para realizar la evaluación por juicio de expertos; dichos atributos se observan en la tabla 5.

Tabla 5

Atributos de calidad según ISO 25010

Dimensión de calidad	Atributo de calidad
Adecuación funcional	Complitud funcional
	Corrección funcional
	Pertinencia funcional
Eficiencia de desempeño	Comportamiento del tiempo
	Utilización de recursos
	Capacidad
Compatibilidad	Coexistencia
	Interoperabilidad

(*continúa*)

(*continuación*)

Usabilidad	Entendimiento
	Aprendizaje
	Operatividad
	Protección a errores
	Atractivilidad
	Accesibilidad
Portabilidad	Adaptabilidad
	Facilidad de instalación
	Reemplazabilidad
Productividad	El framework o librería contribuye en la productividad del desarrollo de gráficos
Satisfacción	Satisfacción del programador (experto) en el uso del framework o librería

Elaboración propia

Para la evaluación de juicio por expertos, se consideró la recomendación de Jakob Nielsen (2000), quien sostiene que la evaluación de usabilidad web solo debe realizarse con cinco usuarios, ya que con más población se hace una inversión innecesaria. Sin embargo, en la presente investigación se consideró seis expertos para la evaluación de EchartsTag, en relación al instrumento que está en función a los atributos de calidad de la ISO 25 010 que se observa en la tabla 5. Asimismo, antes de ello también se consideró cierto perfil para la selección de expertos. Dicha evaluación fue de forma virtual, en función a los demos construidos, y se obtuvieron resultados favorables como se aprecia en la figura 16, donde el 83 % de los expertos consideran que EchartsTag cumple con los atributos de calidad de *software* por lo que sostienen que están muy de acuerdo; mientras que el 16,67 % sostiene que está de acuerdo; por consiguiente, se concluye que el 100 % de expertos están de acuerdo con el cumplimiento de los atributos de calidad de EchartsTag al evaluar las dimensiones de adecuación funcional, eficiencia de desempeño, compatibilidad, usabilidad, portabilidad, productividad y satisfacción; para esta etapa se recomienda seleccionar adecuadamente a los expertos que tenga el perfil según lo que se desea evaluar; para este caso, por ejemplo, deben ser profesionales que hayan trabajado con Java; por ser la herramienta construida para esa comunidad.

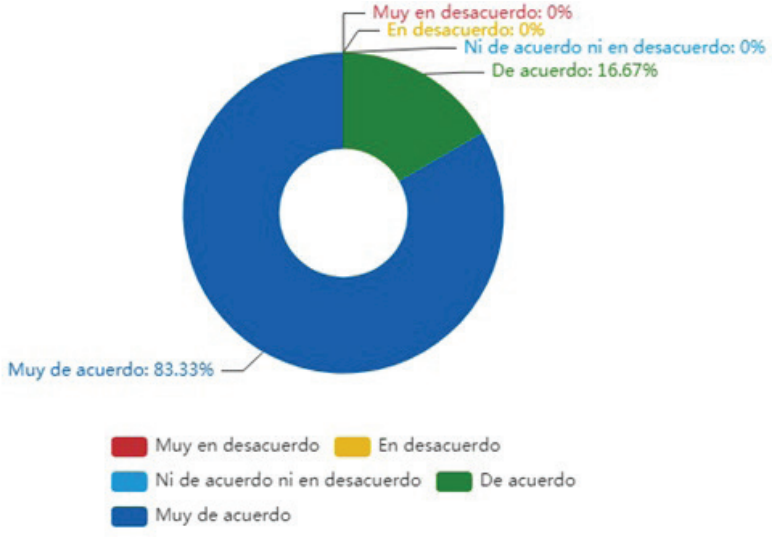

Figura 16. Resultado de la evaluación de atributos de calidad de EchartsTag

Elaboración propia

4.2.3 EchartsTag – Análisis comparativo

Generalmente la información que se usa para la presentación de visualizaciones de gráficos o *charts* ya son datos previamente procesados; por lo que estas no son en grandes cantidades; sin embargo, el soporte de grandes cantidades depende del tipo de gráfico; por ejemplo, se hicieron pruebas con Área Chart y no presentó ningún inconveniente con más de cinco mil registros, al igual que las otras herramientas con las que se hizo comparación; sin embargo, en los casos se trabajó con Bar Chart, y Pie Chart. En la tabla 6 se muestran los resultados obtenidos del experimento realizado bajo las mismas condiciones, como se puede apreciar en los videos donde se observa que EchartsTag obtuvo el menor tiempo en desarrollo de 14,17 minutos en promedio sobre un par de casos, precisando el control de tiempo se hizo mediante un cronómetro digital, seguido por HighCharts y JFreeChart. Por otro lado, también se hizo el control del tiempo sobre la demora o latencia en la visualización de la gráfica, donde destacan Chart.js y HighCharts, seguidos de EchartsTag y FusionCharts; las pruebas se hicieron en la misma máquina con una la capacidad de 32 GB de memoria RAM y CPU de 3,6GHz Intel Xeon.

Tabla 6
Resultado de la evaluación de atributos de calidad de EchartsTag

Frameworks/ Librerias	Tipo – gráfico	Tiempo en desarrollo (min)	Promedio (min)	Evidencia	Promedio tiempo de ejecución (ms)	Promedio (ms)
EchartsTag	PieChart	10	14,17	https://www.youtube.com/watch?v=XCODV0CAG2g	3	3,5
	BarChart	18,34			4	
FusionCharts	PieChart	11	22,775	https://www.youtube.com/watch?v=xwXoWtrfXzA	3	3,5
	BarChart	34,55			4	
JFreeChart	PieChart	14	14,5	https://www.youtube.com/watch?v=geuuwngQFlQ	20	25
	BarChart	15			30	
Chart.js	PieChart	26,58	33,29	https://www.youtube.com/watch?v=UUhAjEUGICE	2	2,75
	BarChart	40			3,5	
HighCharts	PieChart	9	14,22	https://www.youtube.com/watch?v=5C9bpEX4YBI	2,5	3
	BarChart	19,44			3,5	

Elaboración propia

En la tabla 7 se aprecian los resultados en que EchartsTag obtiene un mayor puntaje sobre los criterios planteados; por lo que sería una buena propuesta optar por esta alternativa, ya que FusionCharts e HighCharts son las que más se asemejan en cuanto a interactividad; sin embargo, implican costos por su uso comercial, además de requerir conocimientos en JavaScript. Por otro lado, podría ser Charts.js, sin embargo implica mayor tiempo en desarrollo y no ofrece una información más clara en cuanto a su manual como las otras alternativas, mas no se podría optar por JFreeChart ya que no son interactivas; mientras que EchartsTag internamente usa ECharts desarrollada en JavaScript, para que sus gráficos sean interactivos sin tener conocimiento de JavaScript.

Tabla 7
Análisis de otras características para elegir un framework/librería para charts

Frameworks/ Librerías	No requiere conocimiento en JavaScript	Gráfico interactivo	Soporte Canvas	Soporte de gráficos 3D (3 dimensiones)	Código open source	No requiere pago (uso comercial)	Puntaje
EchartsTag	1	1	1	1	1	1	6
FusionCharts	0.5	1	1	0	0	0.5	3
JFreeChart	1	0	0	0	1	1	3
Chart.js	0	1	1	0	1	1	4
HighCharts	0	1	0	1	0	0	2

Elaboración propia

5. CONCLUSIONES

EchartsTag es una herramienta alternativa para la visualización de charts en el ecosistema Java web y para el desarrollo primeramente se hizo una revisión de literatura sobre herramientas de *charts* en Java en la que se identificaron las principales visualizaciones de *charts*, las cuales se convirtieron en historias de usuarios en relación con la metodología adoptada, En función de ello fue construida usando los conceptos de *taglibs* y *webjars* en Java para la creación de las etiquetas que puedan ser referenciados en el lado del usuario, facilitando el desarrollo de visualizaciones de *charts* mediante la instancia de EchartsTag. Así mismo, se sometió la librería a la evaluación de calidad de código en SonarQube obteniendo resultados favorables de nivel A en cuanto a la fiabilidad, seguridad y mantenibilidad; por otro lado también se evaluó las otras dimensiones de calidad que contempla la ISO 25 010, donde se obtuvo resultados favorables a través del juicio de expertos, respecto a los atributos de calidad que contempla EchartsTag, se obtuvo una calificación favorable de parte del 100 % de expertos que evaluaron a EchartsTag. Con ello podemos concluir que la herramienta construida tiene ciertos criterios de calidad, por lo que se puede tener la confianza al ser instanciada en cualquier proyecto propio de Java web; así mismo, está disponible en los repositorios o gestores de dependencia en JitPack para que pueda ser referenciada por cualquier desarrollador de la comunidad Java en el desarrollo de aplicaciones web. Además, se hizo un análisis comparativo en el que EchartsTag destaca en cuanto al tiempo promedio de desarrollo en la evaluación de un par de casos; así mismo, se evaluaron algunas otras características para elegir una herramienta para *charts* donde también destaca EchartsTag.

6. TRABAJOS FUTUROS

Un aspecto importante a incorporar en la presente investigación es el soporte de *charts* para proyectos de tipo JSF. Considerando que tiene soporte para otros tipos de proyectos de Java, a excepción de JSF; por ello se recomienda ampliar la funcionalidad de EchartsTag para proyectos de tipo JSF aplicando facelet-taglib.

Por otro lado, también es importante recordar que contempla las visualizaciones (*charts*) básicas; por lo que se recomienda ampliar las funcionalidades para otros tipos de *charts* como calendarios, mapas, etc.

Asimismo, ampliar el soporte de internacionalización para diferentes idiomas, utilizando estándares de internacionalización i18n compatible.

REFERENCIAS

Alimam, H. (2017). ChartistJSF: gráficos receptivos altamente personalizables para JavaServer Faces. https://github.com/hatemalimam/ChartistJSF

Burtini, G., Fazackerley, S., y Lawrence, R. (2013). Reducing data transfer for charts on adaptive web sites. *Proceedings of the ACM Symposium on Applied Computing (SAC '13)*, 865–867. https://doi.org/10.1145/2480362.2480528

Gilbert, D. (2019). *JFreeChart.* http://www.jfree.org/

Goto, T., Tsuchida, K., y Nishino, T. (2014). EPISODE: An extreme programming method for innovative software based on systems design. *Proceedings - 2014 IIAI 3rd International Conference on Advanced Applied Informatics, IIAI-AAI*, 780–784. https://doi.org/10.1109/IIAI-AAI.2014.157

Hua, E. C., Nen, V. Y., Tee, F. S., y Ann, O. C. (2018). Pigeon-chart: A customized HTML element for data visualization in data-driven web application using angularjs, highcharts, underscorejs and PHP. *IEEE 3rd International Conference on Communication and Information Systems, ICCIS 2018*, 247–252. https://doi.org/10.1109/ICOMIS.2018.8644793

Knowm Inc. (2019). XChart - Knowm.org. https://knowm.org/open-source/xchart/xchart-example-code/

Li, D., Mei, H., Shen, Y., Su, S., Zhang, W., Wang, J., Zu, M., y Chen, W. (2018). ECharts: A declarative framework for rapid construction of web-based visualization. Visual Informatics, 2(2), 136–146. https://doi.org/10.1016/j.visinf.2018.04.011

Marashdih, A. W., Zaaba, Z. F., Suwais, K., y Mohd, N. A. (2019). Web application security: An investigation on static analysis with other algorithms to detect cross site scripting. Procedia Computer Science, 161, 1173–1181. https://doi.org/10.1016/j.procs.2019.11.230

Martínez, R. A., Turró, M. R., y Saltiveri, T. G. (2019). Accessible statistical charts for people with low vision and colour vision deficiency. *Proceedings of the XX International Conference on Human Computer Interaction (Interacción '19)*, 1–2. https://doi.org/https://doi.org/10.1145/3335595.3335618

Molina-Ríos, J., y Pedreira-Souto, N. (2020). Comparison of development methodologies in web applications. *Information and Software Technology*, 119, 106238. https://doi.org/10.1016/j.infsof.2019.106238

Nielsen, J. (2000). Why You Only Need to Test with 5 Users. Nielsen Norman Group. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

Oracle. (2019). JavaServer Pages Standard Tag Library. Oracle. https://www.oracle.com/technetwork/java/jstl-137486.html

Oracle. (2021). Fusion Middleware Programming JSP Tag Extensions for Oracle WebLogic Server. Oracle; Oracle. https://docs.oracle.com/cd/E12839_01/web.1111/e13722/tld.htm#TAGLB120

Ostruszka, M., Sakowicz, B., y Napieralski, A. (2007). Universal Web-Based Charts Generator Based on J2EE Platform. *The Experience of Designing and Application of CAD Systems in Microelectronics - Proceedings of the 9th International Conference, CADSM 2007*, 524–528. https://doi.org/10.1109/CADSM.2007.4297638

PrimeTek. (2019). PrimeFaces Ultimate UI Framework for Java EE. https://www.primefaces.org/

Redko, A. (2014). Using JavaFX Charts. Oracle. https://docs.oracle.com/javafx/2/charts/jfxpub-charts.pdf

Robiolo, G., y Santos, S. (2016). Estimación de proyectos de software pequeños basada en el juicio de expertos: un caso de estudio. ASSE 2016, 17º Simposio Argentino de Ingeniería. *Software*, 39–50. http://sedici.unlp.edu.ar/handle/10915/57160

Rodriguez, P. A. (2007). Rediseño del modelo de negocios del Datacenter de telefónica empresas en función de prácticas ITIL [Proyecto de Grado de Magister], Universidad De Chile. Repositorio Académico de la Universidad de Chile]. http://www.tesis.uchile.cl/tesis/uchile/2007/rodriguez_pr/pdf/rodriguez_pr.pdf

Shen, H., Bednarz, T., Nguyen, H., Feng, F., Wyeld, T., Hoek, P. J., y Lo, E. H. S. (2019). Information visualisation methods and techniques: State-of-the-art and future directions. *Journal of Industrial Information Integration*, 16, 100102. https://doi.org/10.1016/j.jii.2019.07.003

Sorapure, M. (2019). Text, Image, Data, Interaction: Understanding Information Visualization. *Computers and Composition*, *54*, 102519. https://doi.org/10.1016/j.compcom.2019.102519

Stack-Overflow, I. (2020). Stack Overflow Developer Survey 2020. https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-professional-developers

Sulova, S. (2019). Models for web applications data analysis. *Proceedings of the 20th International Conference on Computer Systems and Technologies (CompSysTech '19)*, 246–250. https://doi.org/10.1145/3345252.3345262

Wells, D. (2013). Extreme Programming: A Gentle Introduction. http://www.extremeprogramming.org/

Yadav, D., Gupta, D., Singh, D., Kumar, D., y Sharma, U. (2018). Vulnerabilities and security of web applications. *2018 4th International Conference on Computing Communication and Automation, ICCCA*, 1–5. https://doi.org/10.1109/CCAA.2018.8777558

ANEXO A
EVALUACIÓN DE ATRIBUTOS DE NIVEL DE CALIDAD DE *SOFTWARE* (ECHARTSTAG)

Estimado(a) sírvase valorar los atributos de calidad de *software* sobre el EchartsTag. Cada pregunta está en función a los atributos de calidad de acuerdo a la ISO/IEC 25 010 que contempla el modelo de calidad de *software*. La valoración de cada atributo se realiza en los siguientes términos con una (X).

1=Muy en desacuerdo	2=En desacuerdo	3=Ni en desacuerdo ni de acuerdo	4=De acuerdo	5=Muy de acuerdo

DATOS DEL PRODUCTO	DATOS DEL EVALUADOR
SOFTWARE: EchartsTag EchartsTag FABRICANTE: David Mamani Pari By: DESCRIPCIÓN: *Framework*/librería de gráficos (cuadros estadísticos) para el ecosistema Java en entorno web.	NOMBRES Y APELLIDOS Firma y DNI

Dimensión	PREGUNTAS Atributos de calidad	PUNTUACIÓN (1) Muy en desacuerdo	(2) En desacuerdo	(3) Ni de acuerdo ni en desacuerdo	(4) De acuerdo	(5) Muy de acuerdo
ADECUACIÓN FUNCIONAL	1.- Tiene un conjunto de funcionalidades apropiadas para las tareas específicas de gráficos (generar cuadros estadísticos) 2.- EchartsTag contempla el soporte de por lo menos dos idiomas. 3.- EchartsTag contempla el soporte de estilos de presentación (themes=manejo de colores en cuadros estadísticos) 4.- EchartsTag genera los gráficos (cuadros estadísticos) de forma esperada y correcta de acuerdo a los gráficos básicos que contempla. 5.- EchartsTag contempla las funcionalidades básicas requeridas como obligatorios en los Taglibs para su implementación					

(*continúa*)

(*continuación*)

EFICIENCIA DE DESEMPEÑO	6.- El tiempo de respuesta al generar los gráficos 2D (cuadros estadísticos) son aceptables. 7.- El tiempo de respuesta al generar los gráficos 3D (cuadros estadísticos) son aceptables. 8.- La cantidad en consumo de recursos en hardware para la visualización de gráficos es razonable o adecuado. 9.- EchartsTag contempla la totalidad de los requerimientos definidos (tipos de gráficos) como requisitos de usuario antes de la construcción.
COMPATIBILIDAD	10.- EchartsTag es compatible con diversos tipos de proyectos del ecosistema Java en entornos web (Servlets, JSP, framework (Spring))
USABILIDAD	11.- EchartsTag es fácil de entender, reconocer la estructura y la lógica de su aplicabilidad. 12.- EchartsTag es fácil de aprender a usar. 13.- EchartsTag es fácil de operar o implementar. 14.- EchartsTag contempla el control de errores a nivel de desarrollo para que los gráficos se generen correctamente. 15.- Es atractivo el diseño de gráficos (cuadros estadísticos)
PORTABILIDAD	16.- EchartsTag se adapta a cualquier sistema operativo en cuanto a su funcionamiento dentro del ecosistema Java en entorno web. 17.- La instalación de EchartsTag es relativamente fácil con el gestor de paquetes Maven. 18.- Al momento de usar EchartsTag es fácil de cambiar de versiones e incluso de ser reemplazado por otro del mismo tipo.
PRODUCTIVIDAD	19- Considera que EchartsTag ayuda en la productividad de desarrollo de software relacionada a la visualización de gráficos (cuadros estadísticos) 20.- EchartsTag permite generar de forma más rápida los gráficos (cuadros estadísticos) en el ecosistema Java en entorno web.

(*continúa*)

(*continuación*)

SATISFACCIÓN	21.- Está satisfecho con el desarrollo (construcción) de EchartsTag 22.- Está satisfecho al probar o implementar los ejemplos (demos) de EchartsTag