Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Sistema Auditoría de Datos: Módulo de Gráficos v2.0

Autora: Ana del Carmen Espinosa Robert

Tutores: MsC. Yadira Barroso Rodríguez

Ing. Anyer Gámez Guedes

“Año 57 de la Revolución”

2014-2015

La mayor parte de las cosas por las que nos esforzamos no cambian mucho,
y solo una pequeña parte de esas cosas, son las que producen los objetivos deseados.
Por eso, en nuestra búsqueda del éxito o la felicidad,
lo más importante no es hacer más o menos cosas,
sino enfocarse en aquellas que proporcionan los mejores resultados.

A: Rebeca, Yadira y Anyer.

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis que tiene por título Sistema Auditoría de Datos: Módulo de Gráficos v2.0 y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ____ días del mes de ____del año ____.

Ana del Carmen Espinosa Robert

_____

Autora

MsC. Yadira Barroso Rodríguez

_____

Tutor

Ing. Anyer Gámez Guedes

_____

Tutor

DATOS DE CONTACTO

Autora:

Ana Del Carmen Espinosa Robert

Universidad de las Ciencias Informáticas

La Habana, Cuba

e-mail: acespinosa@estudiantes.uci.cu

Tutores:

MsC. Yadira Barroso Rodríguez

Universidad de las Ciencias Informáticas

La Habana, Cuba

e-mail: ybarroso@uci.cu

Ing. Anyer Gámez Guedes

Universidad de las Ciencias Informáticas

La Habana, Cuba

e-mail: agguedes@uci.cu

RESUMEN

La auditoría es el proceso mediante el cual se contrata a una empresa o a un auditor para recopilar información contable. En el Centro de Tecnologías y Gestión de Datos (DATEC) se desarrolló el Sistema Auditoría de Datos (AUDAT), para facilitar a los especialistas de la Contraloría General de la República de Cuba (CGR) detectar incidencias en la manipulación de bases de datos de sistemas que ellos auditan. Actualmente su módulo de Gráficos contiene gráficos de barras, sectores, líneas y área para comparar información a través de la representación de los cambios que se producen en los datos. A pesar de las ventajas que ofrece también posee limitantes, a partir de las cuales se realizó la presente investigación. La misma tiene como objetivo desarrollar la versión 2.0 del módulo de Gráficos en el sistema informático AUDAT para profundizar en el análisis de la información a través de la representación gráfica de los datos. Para orientar la investigación se aplicaron métodos teóricos y empíricos que permitieron proponer nuevas alternativas para medir otros aspectos necesarios en la labor que desempeñan los gráficos, así como la selección de las tecnologías y herramientas necesarias para su generación. Para guiar el proceso de desarrollo se seleccionó como metodología XP que permitió a través de sus artefactos definir los requisitos funcionales y no funcionales, así como la estructuración del sistema. Como resultado principal se implementó un módulo capaz de generar gráficos que le permitirá a los especialistas de la CGR describir, resumir y analizar información.

Palabras Claves: AUDAT, gráficos, representación de datos

ABSTRACT

Auditing is the process through which a company or an auditor is hired to collect accounting information. In the Data Auditing System (AUDAT) was developed the data Management Technologies Center (DATEC) to facilitate to the specialists of the General Comptroller of the Republic of Cuba (CGR) the detection of incidents in handling databases systems that are audited. Currently, its module contains bar, pie, line and area charts to compare information across the representation of changes that are produced in data. Despite the advantages the module offers, it also has limitations, and from these limitations this research was performed. It has the objective of developing the Chart module v2.0 for AUDAT computer system to deepen in the analysis of the information through the graphical representation of the data. To guide the research theoretical and empirical methods were applied, which allowed proposing new alternatives to measure other aspects required in the work performed by the charts, and the selection of needed technologies and tools for the generation of the charts. XP methodology was selected to guide the development process which allowed through their contrivances to define the functional and non-functional requirements and the structure of the system. As main result, a module capable of generating graphics that allow the CGR specialists to describe, to summarize and to analyze information was implemented.

Keywords: *AUDAT, chart, representation of data.*

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad, los temas referentes a la auditoría atesoran cada vez más relevancia debido a que la información se ha convertido en el activo más importante de las empresas, lo que representa la principal ventaja estratégica para crear sistemas de información con el fin de obtener la mayor productividad y calidad posible de lo que se audita. El proceso de auditoría es un examen exhaustivo y crítico que se realiza para evaluar la validez y la eficiencia de una sección, un organismo o una entidad. Dicho proceso debe seguir todos los procedimientos y generar toda la documentación y planeación del producto, no solo para salir exitosos en los procesos de auditorías externas, sino también para asegurar la continuidad y la calidad del producto final.

En ocasiones, la información proporcionada se decide representar en forma de gráfico y no como tabla, pues los gráficos son un complemento de las tablas que permiten representar la información de modo más expresivo, facilitando su interpretación. Es por eso que también son considerados una poderosa herramienta no solo para describir y resumir la información, sino también para analizarla.

En el Centro de Tecnologías y Gestión de Datos (DATEC) se desarrolló el Sistema Auditoría de Datos (AUDAT), el cual permite a los especialistas de la Contraloría General de la República de Cuba (CGR) la detección de incidencias en la manipulación de bases de datos de sistemas que se auditan. Está compuesto de varios módulos entre los que se encuentran el módulo de Gráficos, donde se representa de manera visual la información que se recoge.

Actualmente, este módulo facilita la comparación de la información estableciendo sus puntos altos y bajos gracias a la presencia de un gráfico de barras. Permite conocer las proporciones relativas de la información auditada, así como su porcentaje debido a que contiene un gráfico circular. Se cuenta también con un gráfico de líneas para representar los cambios que se producen en los valores de determinados datos clasificados generalmente por cantidad o tiempo, así como un gráfico de área para destacar la magnitud de los cambios que se producen en los datos con respecto al tiempo. A pesar de las facilidades que ofrece y como se ha podido apreciar la escasez de gráficos dentro del módulo no proporciona profundidad en la información representada provocando falta de claridad en el análisis de los datos. Entre las limitantes que posee se encuentran también que:

- ✓ No existe una interacción entre el usuario y las imágenes resultantes debido a que no se modifican los gráficos una vez creados ni se muestra el tipo de información a representar de acuerdo al tipo de gráfico seleccionado.
- ✓ No se conoce con exactitud las irregularidades que son detectadas en los datos, para saber hacia dónde dirigir los esfuerzos de una investigación.
- ✓ No es posible mostrar la planificación de cada una de las tareas a realizar por los usuarios, o los recursos empleados para conocer cuánto tardaron en realizarse o cómo están distribuidos.

Por lo anteriormente expuesto se plantea como **problema de la investigación**: ¿cómo contribuir a la representación de la información de los sistemas que son auditados por la Contraloría General de la República de Cuba?, definiéndose como **objeto de estudio** el proceso de representación de la información digital, enmarcado en el **campo de acción** el proceso de representación de la información digital mediante gráficos.

Para darle solución al problema se traza como **objetivo general** desarrollar la versión 2.0 del Módulo de Gráficos para el Sistema Auditoría de Datos, mediante el cual los especialistas de la Contraloría General de la República de Cuba podrán profundizar en el análisis de la información a través de la representación gráfica de los datos y en correspondencia con él se derivan los siguientes **objetivos específicos**:

- ✓ Analizar los conceptos, la metodología, las tecnologías y las herramientas necesarias para el desarrollo del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Diseñar el Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Implementar el Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Comprobar el correcto funcionamiento del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.

Para guiar la lógica de la investigación se enumeran las siguientes preguntas científicas:

1. ¿En qué se fundamenta el proceso de representación de la información digital mediante gráficos?
2. ¿Qué tecnologías, metodología y herramientas seleccionar para desarrollar el Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos?
3. ¿Qué características se deben tener en cuenta para lograr el correcto funcionamiento del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos?
4. ¿Cómo estructurar el proceso de diseño e implementación del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos?

5. ¿Cómo comprobar el correcto funcionamiento del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos?

Con el fin de resolver el problema de la investigación y al mismo tiempo, dar respuesta a las preguntas antes planteadas se enumeran las siguientes tareas de la investigación:

- ✓ Análisis de los conceptos asociados a la representación de la información mediante gráficos.
- ✓ Selección de las tecnologías, metodología y herramientas a utilizar en el desarrollo del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Elaboración de las historias de usuario para representar las funcionalidades a desarrollar en el Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Identificación de los requisitos funcionales y no funcionales para el correcto funcionamiento del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Elaboración de las tarjetas Clase-Responsabilidad-Colaboración (CRC) para representar la relación entre las clases del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Elaboración de las tareas de ingenierías para analizar detalladamente las funcionalidades del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Implementación de las funcionalidades del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Definición de la estrategia de prueba a partir de la metodología propuesta.
- ✓ Aplicación de las técnicas de prueba para comprobar el funcionamiento del módulo implementado.

Para guiar la investigación se aplicaron **métodos teóricos** y **empíricos**. Los **métodos empíricos** se utilizan para descubrir y acumular un conjunto de hechos y datos como base para dar respuesta a las preguntas científicas realizadas en una investigación, pero no son suficientes para profundizar en las relaciones esenciales que se dan en los procesos. Los **métodos teóricos** posibilitan, a partir de los resultados obtenidos por medio de los **métodos empíricos**, sistematizarlos, analizarlos, explicarlos, descubrir qué tienen en común para llegar a conclusiones confiables que permitan resolver el problema.

De los **métodos empíricos** se aplicó la observación **(ver Anexo 1)** para determinar cuán fácil o engorroso es la utilización de la biblioteca empleada en el *software*, así como la posibilidad de manejar otras que pudieran ser aplicables a la solución.

De los **métodos teóricos** se empleó el análisis-síntesis para investigar y examinar diversos gráficos ya existentes y prediseñados el cual favoreció un análisis de cada uno a través de sus partes: las funciones que cumplen y sus características. Este proceso permitió determinar cuáles proponer como nuevas alternativas que permitan a los auditores medir otros aspectos necesarios en la labor que desempeñan, resumiendo todo el estudio en las nuevas propuestas.

Se aplicó también el método inductivo-deductivo con el objetivo de llegar a conclusiones a partir del estudio y la comparación de las bibliotecas existentes junto a la actual en la confección de gráficos interactivos en el sistema AUDAT, lo que permitió proponer la incorporación de nuevos gráficos que también contribuyan a la solución del problema.

El siguiente trabajo de diploma está estructurado en tres capítulos:

CAPÍTULO I: FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN

Este capítulo describe aspectos teóricos muy elementales referentes a los gráficos y sus características para argumentar la investigación propuesta. A partir del análisis de cada uno de los aspectos teóricos descritos, se hace una propuesta de las herramientas y tecnologías a utilizar en el sistema AUDAT, con el objetivo de dar respuesta a la situación problemática planteada.

CAPÍTULO II: CARACTERÍSTICAS DEL MÓDULO

Este capítulo comprende las etapas de análisis y diseño de la solución. Se muestra el modelo del dominio que le posibilita al usuario un mayor entendimiento del contexto en el que se desarrolla el problema. Se realiza una propuesta de la solución, en la que se definen las principales funcionalidades a desarrollar para darle cumplimiento a lo requerido por el cliente por medio de las historias de usuario, así como la generación de artefactos correspondientes a la etapa de diseño que propone la metodología XP.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO

De acuerdo a la metodología propuesta, la fase de codificación durante el proceso de desarrollo del *software* es importante, debido a que en ella se desarrolla cada funcionalidad del sistema y se comprueba que se realizó exactamente lo que estaba propuesto. Este capítulo muestra también los estándares de codificación aplicados y describe las pruebas realizadas al módulo, con el objetivo de comprobar el cumplimiento de las tareas de ingeniería planificadas para el desarrollo del mismo.

CAPÍTULO I: FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN

Introducción

En el presente capítulo se describen aspectos teóricos muy elementales, referentes a los gráficos y sus características para argumentar la presente investigación. A partir del análisis de cada uno de los aspectos teóricos descritos, se hace una propuesta de las herramientas y tecnologías a utilizar en el sistema AUDAT, con el objetivo de dar respuesta a la situación problemática planteada.

1.1 Conceptos asociados a la investigación presentada

Auditoría

El término auditoría se puede encontrar con varias definiciones pero siempre con el mismo objetivo. Una de ellas hace alusión al concepto de auditoría en términos más amplios, refiriéndose a los pasos que debe hacer cada auditor para revisar y evaluar determinadas “especificaciones de la compañía y los sectores en los que se trabaja”. (1)

También se pueden encontrar otras definiciones como “un proceso sistemático para obtener y evaluar de manera objetiva las evidencias relacionadas con informes sobre actividades económicas y otros acontecimientos relacionados, cuyo fin consiste en determinar el grado de correspondencia del contenido informativo con las evidencias que le dieron origen, así como establecer si dichos informes se han elaborado observando los principios establecidos para el caso”. (2)

De manera general, se asume como definición que la auditoría es un estudio sistemático que se le realiza a un producto, proceso o entidad y que se debe documentar para evaluar el estado en que se encuentra. A partir de entonces, se realizan valoraciones para determinar la importancia o eficiencia que posee lo que se audita: si cumple o no con lo que está establecido o con las condiciones con que se prescribe. Varios autores han coincidido en que se pueden clasificar en distintas clases como las que a continuación se describen:

- ✓ Personal que la desempeña
    - Auditoría Interna
    - Auditoría Externa
- ✓ Objeto de análisis
    - Auditoría de Gestión

- Auditoría Operativa
- Auditoría Financiera
- Auditoría Contable
- Auditoría Informática

✓ Profundidad del análisis
- Auditoría Total
- Auditoría Parcial

✓ Frecuencia del análisis
- Auditoría Continua
- Auditoría Ocasional

Una vez determinado los tipos de auditorías de acuerdo a la clasificaciones descritas: según el objeto de análisis y acotando la investigación al área específica en que se desarrolla, se procede a analizar el término de auditoría informática y su repercusión en el presente trabajo de diploma.

Auditoría informática

A continuación se muestra el concepto de auditoría informática dado por algunos estudiosos en el tema:
Gonzalo Alonso-Rivas, máster en Economía y Dirección de Empresas, define la auditoría informática como “un examen metódico del servicio informático o de un sistema informático en particular, realizado de una forma puntual y de modo discontinuo, a instancias de la dirección, con la intención de ayudar a mejorar conceptos como la seguridad, la eficacia y la rentabilidad del servicio o del sistema que resultan auditados”. (3)

El máster en Administración José Antonio Echenique define la auditoría como “el conjunto de procedimientos y técnicas que sirven para evaluar y controlar un sistema informático como su nombre lo indica, con el fin de constatar si sus actividades son correctas y de acuerdo a la normativa informática establecidas en la organización”. (4)

Luego de analizar los conceptos anteriores, la autora de la presente investigación considera que la auditoría informática es el estudio que se realiza para comprobar la fiabilidad de una herramienta informática y la utilización de ella en una organización.

Se conoce que un sistema es auditable cuando le permite al auditor la revisión de su actividad en cualquier momento, por lo que debe demostrar que funciona de acuerdo a las especificaciones descritas, obedeciendo las respuestas de control, de modo que se utilice como ha sido creado conforme con las normas y las buenas prácticas. (5) Para obtener este criterio los sistemas deben ser construidos con componentes auditables que pueden ser la Auditoría Física, la Auditoría de Calidad, la Auditoría de la Seguridad, la Auditoría de Redes y la Auditoría a las Bases de Datos que es el componente que ocupa la presente investigación.

La Auditoría de Datos se realiza para determinar si los mecanismos de seguridad de una empresa son adecuados y cumplen determinados objetivos o estrategias para proteger la información que se almacena. Con este fin se establecen los cambios que se deben efectuar para el cumplimento de los mismos. (6)

1.2 Representación de datos

Una vez generada la documentación en el proceso de auditoría el auditor necesita representar la información obtenida para analizar los datos detectados y tomar decisiones basadas en tal análisis. Una de las ramas que mayor soporte provee es la estadística que se conoce como "…un conjunto de técnicas para analizar, describir e interpretar datos recolectados de un fenómeno de interés con el fin de tomar decisiones y obtener conclusiones." (7) La representación de datos constituye uno de los aspectos más empleados de la estadística descriptiva. Existen varias maneras de representar datos, entre las más comunes se encuentran la **representación tabular** y **la representación gráfica.**

La **representación tabular** presenta toda la información de modo esquemático. Muestra los datos a través de un conjunto de filas y de columnas que responden a un ordenamiento lógico y se prepara para los cálculos posteriores. Sin embargo, la **representación gráfica** transmite esa información de modo más expresivo. Permite a simple vista, entender de qué se trata lo mostrado, observar sus características más importantes y sacar alguna conclusión sobre el comportamiento de la muestra donde se está realizando el estudio.

Se denomina gráfico o **representación gráfica** a aquellas imágenes que permiten representar información cuantitativa combinando la utilización de sombreado, colores, puntos, líneas, símbolos, números, texto y un sistema de referencia (coordenadas). (9) Los gráficos transmiten de forma inmediata una idea general sobre los principales aspectos de la información por lo que debe construirse de manera sencilla, de fácil interpretación y suministrar valores aproximados.

Como se explicó en la introducción, el Módulo de Gráficos v1.0 del sistema informático AUDAT contiene solamente cuatro gráficos (barras, sectores, línea y área) lo que imposibilita realizar una interpretación más completa de la información. Por este motivo se decide incluir nuevos gráficos en la nueva versión del módulo y los cuatro que ya existían serán redefinidos (de acuerdo a sus características) durante la implementación. A continuación se realiza una breve descripción de todos los gráficos **(Ver Anexo 2)** que serán parte de la solución propuesta.

Gráfico de barras

Un gráfico de barras se utiliza para representar datos cualitativos (ordinales o nominales) o cuantitativos discretos y representan tantas barras como categorías tenga la variable. (8) Este tipo de gráfico permite realizar comparaciones individuales entre dos o más elementos independientemente de la clasificación que exista, gracias a su facilidad de interpretación y construcción. **(Ver figura 15)**

Gráfico de sectores

Se conoce también como gráfico de torta, de pastel o circular y constituyen recursos estadísticos que se emplean para representar porcentajes y proporciones de datos cualitativos. (8) Se utilizan en aquellos casos donde interesa mostrar el número de veces que se da una característica o atributo, de tal manera que se pueda visualizar mejor la proporción en que aparece esa característica de acuerdo al total. **(Ver figura 16)**

Gráfico de línea

Son utilizados principalmente para representar datos cuantitativos generalmente clasificados por cantidad o tiempo. (9) Un gráfico lineal se utiliza frecuentemente para resaltar la manera en que cambian los datos de situaciones que ocurren en períodos sucesivos. **(Ver figura 17)**

Gráfico de área

Los gráficos de áreas permiten generar una representación visual del comportamiento de datos cuantitativos discretos. (9) Los gráficos de área son muy similares a los gráficos de líneas, muy útiles para expresar los cambios que se producen en los valores entre las distintas categorías de datos. Se utilizan para mostrar las tendencias de los valores a lo largo del tiempo o entre categorías. **(Ver figura 18)**

Gráfico de dispersión

Un gráfico de dispersión permite representar el grado de intensidad de la relación entre dos variables cuantitativas. Esta relación puede ser entre un efecto y una de las supuestas causas que lo producen o la

relación entre dos causas que provocan un mismo efecto. (10) Este tipo de gráfico constituye una de las herramientas básicas para el control de la calidad. **(Ver figura 19)**

Gráfico de histograma

El histograma es la representación gráfica de una variable cuantitativa en forma de barras y como parte de un gráfico de barras permite la comparación de los resultados de un proceso. (10) Se utiliza para analizar una variable continua, como las edades o la altura de una muestra, y por comodidad, sus valores se agrupan en clases o valores continuos. En el caso de las encuestas permite revelar datos facilitando su lectura y entendimiento como una forma más cómoda y dinámica de mostrar los resultados de un procedimiento específico. **(Ver figura 20)**

Gráfico de polígono de frecuencias

El polígono de frecuencias es un gráfico que se construye a partir de la unión de los puntos medios de las columnas de un histograma de frecuencias. (10) La diferencia fundamental entre ambos es que a éste último se añaden (opcionalmente) dos clases con frecuencia cero: una antes de la primera clase con datos y otra después de la última y como resultado se obtiene la forma de un polígono. Otra diferencia es que los polígonos de frecuencias para datos agrupados se construyen a partir de la marca de clase que coincide con el punto medio de cada columna del histograma. **(Ver figura 21)**

Diagrama de Gantt

El diagrama de Gantt es una herramienta de planificación de actividades que permite observar el desarrollo de una secuencia de acciones a lo largo del tiempo a través de una representación gráfica mediante barras horizontales, de un plan de trabajo o de un calendario de actividades. (11) **(Ver figura 22)**

El diagrama de Gantt dentro del proceso de auditoría se utiliza para la asignación de recursos. Permite desarrollar un plan detallado en el que se describen los pasos a seguir para la realización de cada tarea y estimar el tiempo de manera realista teniendo en cuenta el personal disponible. Una vez desarrollado el plan se compara el progreso real con lo planificado. Luego se ajusta el plan a lo real y se toman las decisiones correctivas: si al comparar real con lo planeado se determinan avances o retrasos se deben reasignar tareas.

Gráfico de burbujas

El gráfico de burbujas es una variación del gráfico de dispersión donde los puntos son reemplazados por burbujas. En una tabla de datos el tamaño de las burbujas corresponde al tercer valor que equivale a la

intersección de los dos ejes de coordenadas. Se magnifica el impacto cuando las burbujas varían por color y tamaño, lo que proporciona significado sobre los datos. (12) Los gráficos de burbujas son utilizados frecuentemente para representar información financiera pues los diferentes tamaños de las burbujas enfatizan adecuadamente los diferentes valores. **(Ver figura 23)**

Gráfico de Pareto

El gráfico de Pareto es una herramienta de análisis que ayuda a tomar decisiones en función de prioridades. Se basa en un principio enunciado por Wilfredo Pareto que plantea que “el 80% de los problemas se pueden solucionar si se eliminan el 20% de las causas que los originan". Es un caso particular del gráfico de barras que se utiliza para identificar y dar prioridad a los problemas más significativos de un proceso. (11) **(Ver figura 24)**

Una vez analizados los conceptos más importantes referentes al tema de la representación de la información durante el proceso de auditoría, así como la selección de los gráficos en función de las necesidades del Sistema Auditoría de Datos, se procede a la selección de la metodología a seguir para guiar el proceso de desarrollo de la nueva versión del módulo.

1.3 Metodología para el desarrollo del *software*

Para guiar el proceso de desarrollo del módulo se precisa del uso de metodologías ágiles, debido a que se basan en la adaptabilidad ante cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto. Este tipo de metodología guiará el proceso de desarrollo para la solución del problema planteado y tiene como principios que:

- ✓ Los individuos y sus interacciones son más importantes que los procesos y las herramientas.
- ✓ El *software* que funciona es más importante que la documentación exhaustiva.
- ✓ La colaboración con el cliente en lugar de la negociación de contratos.
- ✓ La respuesta ante del cambio en lugar de seguir un plan cerrado.

Una metodología de desarrollo de *software* se utiliza para estructurar, planear y controlar el proceso de desarrollo en sistemas de información y tiene como filosofía el desarrollo de programas con el enfoque del proceso de desarrollo de *software* y en el empleo de herramientas, modelos y métodos para asistir a este proceso. (13)

La metodología de desarrollo especifica los pasos a seguir en cada etapa del desarrollo de un proyecto para obtener un producto eficiente y con calidad. Sin embargo, no todos los proyectos utilizan el mismo proceso de desarrollo. El sistema informático AUDAT fue desarrollado siguiendo la metodología XP, la cual se selecciona atendiendo a las políticas del proyecto.

XP o *Extreme Programming*

Esta metodología se basa en la realimentación continua entre el cliente y el equipo de desarrollo, en la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y está preparada para enfrentar los cambios durante el desarrollo de lo que se implemente. (22)

La programación extrema, a diferencia de las metodologías tradicionales, pone más énfasis en la adaptabilidad que en la previsibilidad. Es por eso que sus defensores consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable durante el desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida de un proyecto es una aproximación mejor y más realista, que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en cada uno.

Características de XP

- ✓ Una característica fundamental de esta metodología es su desarrollo iterativo e incremental, pues en cada iteración se corrigen posibles errores y se le realizan mejoras a cada funcionalidad.
- ✓ Propone pruebas unitarias continuas, frecuentemente repetidas.
- ✓ La frecuente interacción del equipo de desarrollo con el cliente o usuario final.
- ✓ Propone la realización de entregas frecuentes con el objetivo de corregir errores antes de implementar una nueva funcionalidad.
- ✓ Reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad sin modificar su comportamiento.

Fases de XP

Como toda metodología, la programación extrema se rige por fases que aseguran el desarrollo completo del *software* que se implementa. A continuación se describen brevemente cada una de ellas:

I. Planificación del proyecto

En esta primera fase se realiza una recopilación de todos los requisitos del sistema. Es aquí donde se definen las Historias de Usuario (HU), lo que conlleva a una constante interacción entre el

cliente y el equipo de desarrollo. Este período de planificación es para concretar qué es lo que se quiere y cómo se quiere, para lograr los objetivos finales.

II. Diseño

Esta segunda fase propone que hay que lograr diseños simples y sencillos, fáciles de entender e implementar para conseguir reducir el costo de tiempo y esfuerzo en desarrollarlo. Se pretende crear parte del proyecto, algo así como la interfaz que tendrá el cliente con la aplicación. En esta fase surge otro artefacto muy importante que son las tarjetas Clase-Responsabilidad-Colaboración (CRC) que facilitan al desarrollador enfocarse en las clases que va a implementar, estableciendo relaciones entre ellas.

III. Codificación

Como ya se explicó en la etapa de planificación, el cliente forma parte del equipo de desarrollo. Su presencia es un factor clave para el desarrollo del *software* durante todas las fases de esta metodología. En cada HU el cliente especifica detalladamente una funcionalidad y debe estar presente cuando se realicen las pruebas que verifiquen que la historia desarrollada cumple exactamente con la funcionalidad especificada. Durante esta fase, el cliente y el equipo de desarrollado deben mantenerse en constante comunicación para que los desarrolladores puedan implementar todo lo requerido.

IV. Prueba:

Esta etapa se realiza durante todo el proceso de desarrollo de la metodología debido al objetivo fundamental que persigue el uso de XP. Cada vez que se implementa una HU se comprueba por medio de iteraciones que se cumpla exactamente con lo que en ella se define.

1.4 Estudio de las bibliotecas para la generación de gráficos

Hoy en día ha aumentado la utilización de bibliotecas de clases para la representación gráfica de la información. A continuación se realiza un estudio breve de las bibliotecas más comunes que permiten la generación de gráficos.

HighCharts

Es una biblioteca de gráficos implementados en HTML5/*JavaScript* que permite la creación de gráficos interactivos en una aplicación *web*. De acuerdo al estado actual del módulo se analizan las siguientes características:

- ✓ Soporte para gráficos de línea, área, columnas, barras, circulares, de dispersión, patrones angulares, burbuja, diagrama de caja, barras de error, embudo, cascada y tipos de gráficos polares, *area range*, *area spline range*, *column range spline*, *area spline*.
- ✓ Permite generar gráficos en formato .jpg, .png, .pdf y .svg.
- ✓ Se pueden incorporar a la interfaz las *tooltips,* las etiquetas de múltiples ejes, incluyendo de fecha y hora.
- ✓ Se distribuye bajo los términos de la licencia *Creative Communs Attribution-Non Commercial 3.0* (15)

FusionChart

Es una biblioteca de gráficos implementados en HTML5/*JavaScript* que permite la generación de gráficos interactivos para aplicaciones *web*. De acuerdo a las necesidades del proyecto se enumeran las características más importantes de esta biblioteca gráfica:

- ✓ Contiene variedad de gráficos entre los que se encuentran los propuestos como parte de la solución.
- ✓ Incluye una ayuda muy completa (en inglés) con abundantes ejemplos y demostraciones.
- ✓ Se distribuye bajo los términos de la licencia GPL y MIT. (16)

JfreeChart

Es una biblioteca de gráficos para el lenguaje de programación *Java* que permite la creación de gráficos complejos de forma simple. Posee como características las que a continuación se muestran:

- ✓ Compatibilidad con varios gráficos entre los que se encuentran barras, línea, dispersión, sectores, así como la presencia de otros que se adecúan a la solución.
- ✓ Tiene incorporado un dibujador de histogramas.
- ✓ Posibilita colocar varios marcadores en el área de graficar.
- ✓ Dibuja automáticamente las escalas de los ejes y leyendas. (17)

Para realizar la comparación de las bibliotecas anteriores se han tenido en cuenta estos aspectos a partir las políticas definidas en el sistema informático AUDAT.

Tabla 1. Comparación de las bibliotecas para generar gráficos

Bibliotecas	Lenguaje de programación	Cantidad de gráficos que soporta	Formato de exportación	Aplicaciones	Licencia
HighCharts	*JavaScript*	24	.jpg, .png, .pdf, .svg	*web*	MIT
FusionChart	*JavaScript*	Más de 90	.jpeg, .png, .pdf, .svg	*web*	GPL, MIT
JfreeChart	*Java*	22	.jpeg, .png, .pdf, .eps, .svg	*desktop*	GNU/ LGPL

Justificación de la biblioteca seleccionada

HighCharts es una biblioteca que permite la generación de gráficos interactivos para la *web*. Soporta más de 24 gráficos entre los que se encuentran los propuestos en la solución. El sistema informático AUDAT es una aplicación de escritorio que se implementó con el lenguaje de programación *Java*. Teniendo en cuenta este aspecto se decide no emplearla en la solución a pesar de las ventajas que ofrece.

FusionChart es otra biblioteca gráfica que facilita la generación de gráficos. Contiene más de 90 gráficos incluyendo los anteriormente mencionados, así como la exportación a otros formatos como jpeg, .png, .pdf, .svg. Sin embargo, también está implementada con *JavaScript* por lo que se decide no emplearla como parte de la solución por políticas del proyecto.

Partiendo de estas características se selecciona *JfreeChart* como biblioteca para contribuir a la generación de gráficos, pues es la que más se ajusta a las políticas definidas por el sistema informático AUDAT, al estar implementada en *Java* y emplearse para aplicaciones *desktop*. Además dentro de los gráficos que soporta se incluyen los mencionados como parte de la solución. Posee otros formatos de exportación como jpeg, .png, .pdf, .eps, .svg y se distribuye bajo los términos de GNU/ LGPL (*Lesser General Public License*).

1.5 Herramientas y tecnologías para el desarrollo del módulo

***NetBeans IDE* v8.0**

Este entorno de desarrollo permite la creación de aplicaciones multiplataforma, proporciona facilidades y garantías para la migración, cumplimiento de regulaciones y flexibilidad entre plataformas. Para la implementación del módulo se seleccionó la versión 8.0 por poseer entre sus características:

- ✓ Permitir el uso del constructor de interfaz gráfica de manera intuitiva para la creación y edición de componentes de GUIs (interfaces gráficas de usuario), de una paleta en un lienzo o una página HTML para crear prototipos de GUIs, con soporte incorporado para su localización y accesibilidad.
- ✓ Compatibilidad con la biblioteca *JfreeChart* para generar gráficos en *Java*.
- ✓ Puede conectarse a gestores de bases de datos tales como *Oracle*, *MySQL*, *PostgreSQL* y permite autocompletar los nombres de las tablas, realizar consultas y modificaciones, todo ello integrado en el propio IDE de desarrollo. (24)

Justificación del IDE seleccionado

Los módulos desarrollados previamente en el Sistema Auditoría de Datos han definido al *NetBeans IDE v8.0* como herramienta de desarrollo del proyecto. Otra de las ventajas que ofrece es el desarrollo de servicios comunes para aplicaciones de escritorio, permitiéndole al programador enfocarse en la lógica específica de su implementación. Esta decisión también se fundamenta sobre la base de que los desarrolladores del sistema informático AUDAT poseen experiencia en el desarrollo de aplicaciones con este IDE.

Java

Java es un lenguaje que toma mucha de sus sintaxis de C y C++. Permite la escritura del código seguro debido a que no permite el manejo de punteros e incluye gran variedad de bibliotecas. Posee como características:

- ✓ Es multiplataforma, pues los programas escritos en dicho lenguaje pueden ejecutarse igualmente en cualquier sistema operativo.
- ✓ La biblioteca seleccionada *JfreeChart* está implementada en el este lenguaje.
- ✓ Es uno de los lenguajes de programación que posee mayor soporte y recursos lo que facilita su estudio y aprendizaje.

Justificación del lenguaje seleccionado

Los módulos que componen al sistema informático AUDAT se implementaron utilizando *Java* como lenguaje de programación. Por lo tanto, siguiendo las políticas del proyecto se decide su empleo para desarrollar la nueva versión del módulo de Gráficos. Otra consecuencia de su selección se debe a las ventajas que ofrece de soportar las tres características propias del paradigma orientado a objetos: encapsulación, herencia y polimorfismo.

***Visual Paradigm for* UML v8.0**

Otra de las herramientas a seleccionar para la implementación del módulo de Gráficos v2.0 es el *Visual Paradigm* como herramienta CASE[1]. Para la implementación de la solución se seleccionó la versión 8.0 la cual posee como características:

- ✓ Posibilita la generación de código y documentación para entornos integrados de desarrollo como *NetBeans, Eclipse u Oracle*.
- ✓ Facilita el diseño de diagramas que integran el ciclo completo de desarrollo del *software*.
- ✓ Se puede aplicar en una variedad de formas para dar soporte a una metodología de desarrollo de *software*, pero no especifica qué metodología o proceso utilizar. (26)

Justificación de la herramienta seleccionada

A pesar de que XP no genera muchos artefactos, el uso de *Visual Paradigm for* UML v8.0 ayudará a la construcción de un modelo conceptual, así como un diagrama de clases del diseño que favorecerá en gran medida a comprender la solución del problema. Esta herramienta de UML (*Unified Model Language*) que es un conjunto de símbolos y de modos de disponerlos utilizado para modelar el diseño de un *software*.

UML v2.0

Este lenguaje de modelado se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de *software*, principalmente durante el análisis y el diseño del mismo. Para el desarrollo del módulo se empleó este lenguaje en su v2.0 por poseer las siguientes características:

- ✓ Permite expresar de una forma gráfica un sistema de manera que pueda ser entendido por el usuario.

[1] *Computer Aided software Engineering*: Ingeniería de *software* Asistida por Computadora

- ✓ Especifica cuáles son las características de un sistema antes de su construcción.
- ✓ A partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión. (27)

Esta notación se compone de diversos gráficos que se combinan para conformar diagramas con mayor calidad. Es importante destacar que un modelo de UML describe lo que debe hacer un sistema, pero no dice cómo implementarlo.

Conclusiones parciales del capítulo

El análisis de los conceptos relacionados con la representación de datos durante el proceso de auditoría contribuyó a la selección de los gráficos que le darán solución a la situación problemática planteada. Para guiar todo el proceso de desarrollo del módulo se seleccionó la metodología XP, soportada por la herramienta CASE *Visual Paradigm for* UML v8.0 y empleando como lenguaje de modelado UML v2.0 para modelar los artefactos generados. Se seleccionó como lenguaje de programación *Java* utilizando como entorno de desarrollo *NetBeans* v8.0 para la implementación de la nueva versión del módulo de Gráficos, lo que facilitó la selección de la biblioteca *JfreeChart* para la contribuir a la generación de los gráficos establecidos.

CAPÍTULO II: CARACTERÍSTICAS DEL MÓDULO DE GRÁFICOS v2.0

Introducción

El presente capítulo comprende las etapas de análisis y diseño del módulo. Se muestra el modelo del dominio que le facilita al usuario un mayor entendimiento del contexto en que se desarrolla el problema. Se realiza una propuesta de solución en la que se definen las principales funcionalidades a desarrollar, para darle cumplimiento a lo requerido por el cliente a través de las HU, así como la generación de artefactos correspondientes a la etapa de diseño que propone la metodología XP.

2.1 Modelo del dominio

.Según *Craig Larman* “un modelo conceptual explica los conceptos significativos en el dominio de un problema”. (21) La metodología propuesta (XP) no define modelo del dominio, pero teniendo en cuenta que este artefacto permite mostrar de manera visual los principales conceptos que se manejan, se especifica su empleo para ayudar a usuarios, desarrolladores o interesados a utilizar un vocabulario común para poder entender el contexto en que se desarrolla el problema. A continuación se muestra el modelo del dominio correspondiente a la situación actual del módulo:

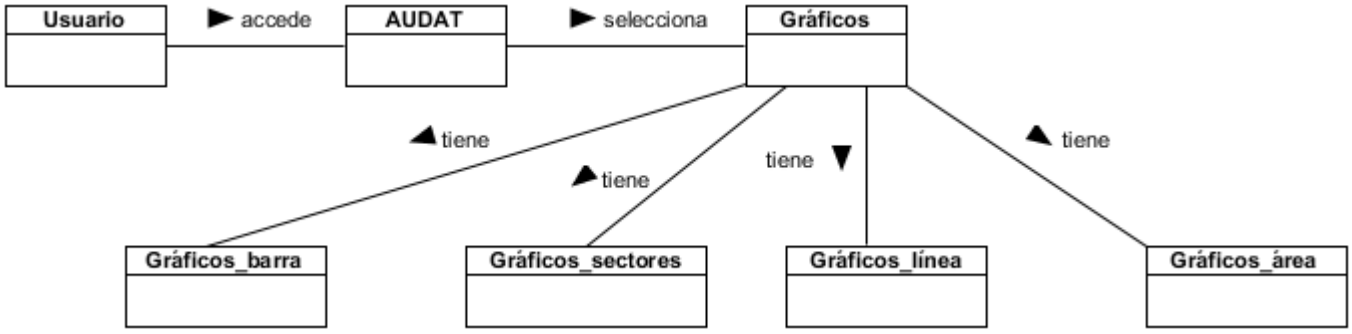

Fig. 1 Modelo de dominio

En la figura se muestra al usuario (que corresponde a un especialista de la CGR) accediendo al sistema informático AUDAT (que representa al sistema donde se almacena la información que se audita). Luego selecciona la opción que corresponde al módulo de Gráficos a través de la barra de menú, o una figura que lo representa, permitiendo mostrar esta información de forma gráfica. Hasta el momento el módulo se compone solamente de los gráficos de barras, sectores, línea y área.

2.2 Solución propuesta

La presente investigación propone como solución a lo anterior desarrollar una nueva versión del módulo de Gráficos del sistema informático AUDAT, para ayudar a los especialistas de la CGR a interpretar la información detectada a través de la generación de gráficos interactivos, lo que permitirá profundizar en el análisis de la misma.

La interactividad de estos gráficos permitirá al auditor modificar sus características después de haber sido creados, ejecutando los cambios en tiempo real. Se podrán observar los resultados de las modificaciones inmediatamente luego de realizadas, lo que constituye una gran ventaja ya que dicho auditor puede ir alterando los parámetros hasta obtener resultados deseados. Dentro de los parámetros que se pueden modificar se encuentran la forma, el tipo de gráfico, los títulos, intercambiar ejes para las variables, ingresar nuevas variables, entre otras.

2.3 Historias de Usuario

Las Historias de Usuario (HU) son breves descripciones del comportamiento de un *software*. Permiten dividir una gran parte de las funcionalidades en partes más pequeñas para lograr la planificación.

Son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios. (22)

Características de las HU

- ✓ Potencian la participación del equipo en la toma de decisiones.
- ✓ Se crean y evolucionan a medida que el proyecto avanza.
- ✓ Son peticiones concretas y pequeñas.
- ✓ Contiene la información imprescindible.
- ✓ Apoyan la cooperación, colaboración y conversación entre los miembros del equipo.

Las HU se definen con el cliente y se emplean para estimar los tiempos de desarrollo del módulo, así como para verificar si se cumple con lo que en ella se especifica una vez que se pasa a la fase de prueba.

Es importante destacar que conducen al proceso de creación de las pruebas de aceptación, las cuales se emplean para verificar que cada historia ha sido implementada correctamente. Las HU deberán poseer al menos los siguientes aspectos:

- ✓ Número: se corresponde con el número asignado a la HU.
- ✓ Nombre de HU: se corresponde con el atributo que contiene el nombre de la HU.
- ✓ Cantidad de modificaciones: cantidad de veces que se registró el cambio.
- ✓ Usuario: el usuario del sistema que utiliza o protagoniza la HU.
- ✓ Prioridad en el negocio: se especifica el nivel de prioridad de la HU en el desarrollo del módulo.
- ✓ Riesgo de desarrollo: se especifica el nivel de riesgo en caso de no realizarse la HU.
- ✓ Puntos estimados**:** Este atributo es una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo. XP define una semana ideal como 5 días hábiles trabajando 40 horas, es decir, 8 horas diarias. Por lo que cuando el valor de dicho atributo es 0.5 equivale a 2 días y medio de trabajo, lo que se traduce en 20 horas.
- ✓ Puntos reales: comprende los aspectos del parámetro anterior, pero en este caso se escribe el tiempo real en el que se realizó la HU.
- ✓ Descripción: se realiza una breve descripción de la HU.

A continuación se muestran las principales historias de usuario generadas durante el desarrollo del módulo.

Tabla 2. HU Importar fichero de datos

Historia de Usuario	
Número: 1	**Nombre:** Importar fichero de datos
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada**:1
Prioridad en el negocio: Muy alta	**Puntos estimados:** 0.8
Riesgo de desarrollo: Muy alto	**Puntos reales:** 0.8

Descripción: Permite generar un fichero a partir de la selección de una fuente de datos desde un directorio para la generación de los gráficos del módulo. Esta fuente de datos puede ser un fichero de datos separados por coma de tipo CSV (.csv), un fichero en formato de *Excel* (.xls), un fichero en formato de *Access* (.mdb), un fichero en formato de dBase (.dbf), un fichero en formato de *SQlite* (.sqlite), o se pueden cargar los datos desde una base datos de *PostgreSQL*, *MySQL* y SQL *Server*.
Observaciones

Tabla 3. HU Generar gráfico de barras

Historia de Usuario	
Número: 2	**Nombre:** Generar gráfico de barras
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada:**1
Prioridad en el negocio: Muy alta	**Puntos estimados:** 1
Riesgo de desarrollo: Alto	**Puntos reales:** 1
Descripción: Permite la creación de un gráfico de barras a partir de la información almacenada y al mismo tiempo se pueden modificar sus características. Esto se logra a través del espacio “Tipo de gráfico” de la interfaz principal del módulo.	
Observaciones	

Tabla 4. HU Exportar gráfico a PDF

Historia de Usuario	
Número: 3	**Nombre:** Exportar gráfico a PDF
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada**:3

Prioridad en el negocio: Media	**Puntos estimados:** 0.5
Riesgo de desarrollo: Medio	**Puntos reales:** 0.5
Descripción: Permite exportar el gráfico en el formato de documento PDF.	
Observaciones	

Para conocer el resto de las HU correspondientes al desarrollo de los gráficos que se implementarán en el módulo **ver anexo 2: Historias de Usuario implementadas.**

Las HU deben detallarse de tal manera, que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará desarrollar cada funcionalidad. Cuando llega el momento de la implementación, el desarrollador debe dialogar directamente con el cliente para obtener todos los detalles. Inicialmente, el equipo de desarrolladores estima el esfuerzo necesario para implementar las historias de usuario y los clientes aprueban los objetivos y los tiempos de entrega.

Sobre la base del análisis de la representación de los conceptos mediante el modelo del dominio, las expectativas del desarrollo del sistema, la necesidad de realizar una correcta captura de los requisitos para evitar demoras en la construcción del *software* y errores en los mismos, se realiza la especificación de requisitos que comprende la descripción completa del desarrollo del sistema.

Un Requisito Funcional (RF) define una función del sistema de *software* o sus componentes y son complementados por los Requisitos no Funcionales (RnF), que especifican los criterios que se deben usar para juzgar el funcionamiento de un sistema. A continuación se enumeran las principales funcionalidades que descritas en las HU anteriores:

RF1. Importar fichero de datos
RF2. Crear gráfico de barras
RF3. Modificar gráfico de barras
RF4. Crear gráfico de sectores
RF5. Modificar gráfico de sectores
RF6. Crear gráfico de línea
RF7. Modificar gráfico de línea

RF8. Crear gráfico de área
RF9. Modificar gráfico de área
RF10. Crear gráfico de histograma
RF11. Modificar gráfico de histograma
RF12. Crear gráfico de dispersión
RF13. Modificar gráfico de dispersión
RF14. Crear gráfico de burbujas
RF15. Modificar gráfico de burbujas
RF16. Crear gráfico polígono de frecuencia
RF17. Modificar gráfico polígono de frecuencia
RF18. Crear gráfico de Pareto
RF19. Modificar gráfico de Pareto
RF20. Crear gráfico de Gantt
RF21. Modificar gráfico de Gantt
RF22. Exportar gráfico a PDF

Para desarrollar el módulo se definió que debe cumplir con los siguientes requisitos no funcionales:

RnF1. **Usabilidad**

El sistema debe presentar facilidades al usuario para el manejo de la información por lo que proveerá una interfaz visual amigable. Para ello presenta una iconografía (en la ventana principal del sistema) que se ajusta con la función que representa: generar gráficos, así como campos de selección para la modificación de los gráficos una vez creados por el usuario.

RnF2. Restricciones de diseño e implementación

El sistema será implementado a través del entorno de desarrollo *NetBeans* v8.0 empleando el lenguaje de programación *Java*.

RnF3. Apariencia e interfaz externa

La interfaz el sistema se alinea a la gama de los productos de la marca de XEDRO: un degradado de color suave (230, 224, 236), tipo de letra SansSerif, sin cursiva, de tamaño 12. La interfaz de la solución se dividirá en dos paneles, una para la selección y modificación de los gráficos y la otra parte para visualizarlos, de manera que el usuario final pueda familiarizarse sin dificultad con la nueva versión del módulo.

RnF4. *Hardware*

Las estaciones de trabajo deben cumplir con los siguientes requisitos de *hardware*:

- ✓ Computadora *Pentium* IV o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 1 GB.
- ✓ Un mínimo de 10GB de espacio en disco para instalar el sistema.

RnF5. *Software*

Las estaciones de trabajo donde se instalará la aplicación deben cumplir con los siguientes requisitos:

- ✓ Sistema Operativo:
  GNU/*Linux* preferentemente *Ubuntu* GNU/*Linux* 14.10 o superior, *Debian* 4 GNU/*Linux* o superior, *Windows* 7 o superior.
- ✓ Open-JDK.7 o superior, máquina virtual de *Java.*

Para la realización del presente módulo se utilizarán las semanas como medida para la estimación por esfuerzo establecida por el programador. A continuación se muestra la tabla de estimación por esfuerzos:

Tabla 5. Estimación por esfuerzos

No.	Historia de usuario	Estimación (por semanas)
1	Importar fichero de datos	0.8
2	Crear gráfico de barras	1
3	Crear gráfico de sectores	1
4	Crear gráfico de línea	1
5	Crear gráfico de área	1
6	Crear gráfico de histograma	1
7	Crear gráfico de polígono de frecuencia	1
8	Crear gráfico de dispersión	1

9	Crear gráfico de burbujas	1
10	Crear gráfico de Pareto	1
11	Crear gráfico de Gantt	1
12	Exportar gráfico a PDF	0.5

2.4 Plan de iteraciones

Las iteraciones son fases o etapas de la implementación donde se obtienen resultados en un tiempo estimado. En el plan de iteraciones se especifican detalladamente el orden de desarrollo de las historias de usuario en cada iteración conjuntamente con la duración de las mismas.

Iteración 1: En esta iteración se implementan las historias de usuario que tienen prioridad alta en el negocio. Teniendo en cuenta que la versión anterior del módulo estaba compuesta por los gráficos de barras, sectores, área y línea se les asigna prioridad Muy alta a las HU 1, 2, 3, 4 y 5, las cuales comprenden la realización de los requisitos mencionados incluyendo el requisito Importar fichero de datos, debido a que forma parte del flujo de la solución. Con esta iteración se obtiene la primera versión del módulo permitiendo darle vista al cliente y favoreciendo la retroalimentación en el grupo de trabajo.

Iteración 2: En esta iteración se realiza la implementación de las HU 6, 7, 8 y 9. Permite corregir errores o inconformidades del cliente con las historias de usuario desarrolladas en la iteración anterior. De esta forma se obtiene la segunda versión del sistema. Al igual que la primera iteración, también se le debe mostrar al cliente para que se evalúen las aceptaciones de éste con lo implementado.

Iteración 3: En esta iteración se realiza la implementación del resto de las HU 10, 11 y 12 corrigiéndose también los errores de las iteraciones anteriores.

Una vez estimado en días el tiempo que demora implementar cada historia de usuario, se conforma el plan de duración de las iteraciones. El objetivo de este artefacto es especificar detalladamente el orden de desarrollo de las HU dentro de cada iteración y la duración de cada una de ellas.

Tabla 6. Plan de duración de las iteraciones

Iteraciones	Orden de las HU a implementar	Estimación	Duración (por semanas)
Iteración 1	Importar fichero de datos	0.8	4.8
	Crear gráfico de barras	1	
	Crear gráfico de sectores	1	
	Crear gráfico de línea	1	
	Crear gráfico de área	1	
Iteración 2	Crear gráfico de histograma	1	4
	Crear gráfico de dispersión	1	
	Crear gráfico de burbujas	1	
	Crear gráfico de polígono de frecuencia	1	
Iteración 3	Crear gráfico de Pareto	1	2.5
	Crear gráfico de Gantt	1	
	Exportar gráfico a PDF	1	

2.5 Tarjetas CRC

Las tarjetas CRC constituyen una técnica de diseño orientado a objetos cuyo objetivo es hacer, mediante tarjetas, un inventario de las clases que se necesitan para implementar el sistema y la forma en que interactúan unas con otras. De esta manera se facilita el análisis y discusión de las mismas por parte de varios miembros del equipo del proyecto, con el objetivo de que el diseño sea lo más simple posible verificando las especificaciones del sistema. (23)

Tabla 7. Tarjeta CRC Generar gráficos

Clase: Graficadora.java	
Responsabilidad	**Colaboración**
Describe las funcionalidades que comprenden la creación de los gráficos en el sistema**.**	GraficosInteractivos.java TablaBaseDatos.java

Tabla 8. Tarjeta CRC Modificar gráfico

Clase: GraficosInteractivos.java	
Responsabilidad	**Colaboración**
Clase interfaz que permite la interacción del usuario con el módulo para generar los gráficos, así como la posibilidad de modificarlos visualmente de acuerdo a su satisfacción.	JpanelsVariables.java JPVG_Burbujas.java JPVG_Gannt.java JPVG_Histograma.java JPVG_MultiplesVariables.java JPVG_Pareto.java JPVG_XY.java Graficadora.java

2.6 Arquitectura de *software*

Hasta el momento no se conoce una definición única del término arquitectura de *software*. La bibliografía sobre el tema es tan extensa como la cantidad de definiciones que en ella se pueden encontrar. Por lo tanto, en el presente epígrafe se tratará de introducir un concepto sencillo que permita comprender este concepto desde el punto de vista de cada lector.

La arquitectura de *software* "…es una descripción de los subsistemas y los componentes de un sistema informático y las relaciones entre ellos." (24)

Otro de los conceptos que se analizan es la arquitectura de *software* como “la principal estructura de un sistema representada a través de sus componentes, las relaciones que se dan entre ellos y el ambiente y los principios que guían el diseño constantemente. (25)

La autora de la presente investigación considera que la arquitectura de *software* es la estructura de un sistema cuya función es definir los componentes, interacciones y requisitos que deben satisfacer al sistema, así como, las restricciones a las que está sujeto, reglas y decisiones del diseño que gobiernan la estructura junto a los argumentos que justifican las decisiones tomadas .

Una vez aclarado el alcance que puede tomar el término de arquitectura de *software*, resulta de gran interés introducir formalmente otros términos que resultan pilares fundamentales dentro del contexto de la arquitectura, dado que en torno a ellos gira gran parte del estudio que se ha realizado sobre el tema hasta el momento. Tal es el caso de patrón arquitectónico y patrón de diseño. Estos representan, de lo general a lo particular, los niveles de abstracción que componen la arquitectura de *software*.

2.6.1 Patrón arquitectónico

El patrón arquitectónico es el nivel de la arquitectura de *software* en el cual se define la estructura básica de un sistema. Representa una plantilla de construcción que provee un conjunto de subsistemas aportando las normas para su organización. El Sistema Auditoría de Datos está implementado utilizando el patrón arquitectónico Modelo-Vista-Controlador, es por ello que se emplea dicho patrón para desarrollo del módulo. A continuación se describen sus características y funcionamiento.

Modelo-Vista-Controlador (*Model View Controller*)

Este patrón de diseño permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación, la interfaz del usuario y la lógica de control.

Aunque la metodología XP no propone como artefacto el Diagrama de Clases del Diseño (DCD), se propone su elaboración para una mayor comprensión de las tarjetas CRC elaboradas en el epígrafe anterior, en el cual se representan gráficamente las clases y las relaciones, que están representadas en las colaboraciones que existen entre cada una de ellas.

El usuario cuando accede al módulo para representar la información importada a través de los gráficos. La clase **GraficosInteractivos.java** es la clase interfaz del módulo. Permite mostrar al usuario el tipo de gráfico a generar, dependiendo de la información que se obtenga y el tipo de variables que utilice, es por

ello que los otros componentes e interfaces de la vista se muestran a través de ella, como **JPanelsVariables.java**, **JPVG_Bubujas.java**, **JPVG_Gantt.java**, **JPGV_Pareto.java**, **JPVG_XY.java**, **JPVG_Histograma.java** y la clase **JPVG_MultiplesVariables.java**. El paquete de la vista permite al usuario interactuar con el módulo y el diseño se logra a partir del uso de la biblioteca *JfreeChart*.

En el paquete del controlador se encuentran las clases que permiten dar respuesta a las peticiones del usuario realizadas a través de la vista y al mismo tiempo hace una solicitud al modelo para la obtención de los datos. La clase **Graficadora.java** crea un objeto de tipo **TablaBaseDatos.java** para hacer una solicitud a esta clase ubicada en el modelo, de manera que se pueda acceder a la información importada. La clase **Graficadora.java** es la que contiene los métodos para la generación de los gráficos a partir de la información almacenada en la clase **TablaBaseDatos.java**.

En el paquete del modelo se encuentran las clases que permiten la gestión de la información que será representada en el módulo. En este paquete, como se mencionó anteriormente se encuentra la clase **TablaBaseDatos.java** que contiene la información que posteriormente será representada a través de la generación de los gráficos. A continuación se representa el diagrama de clases del diseño de la solución.

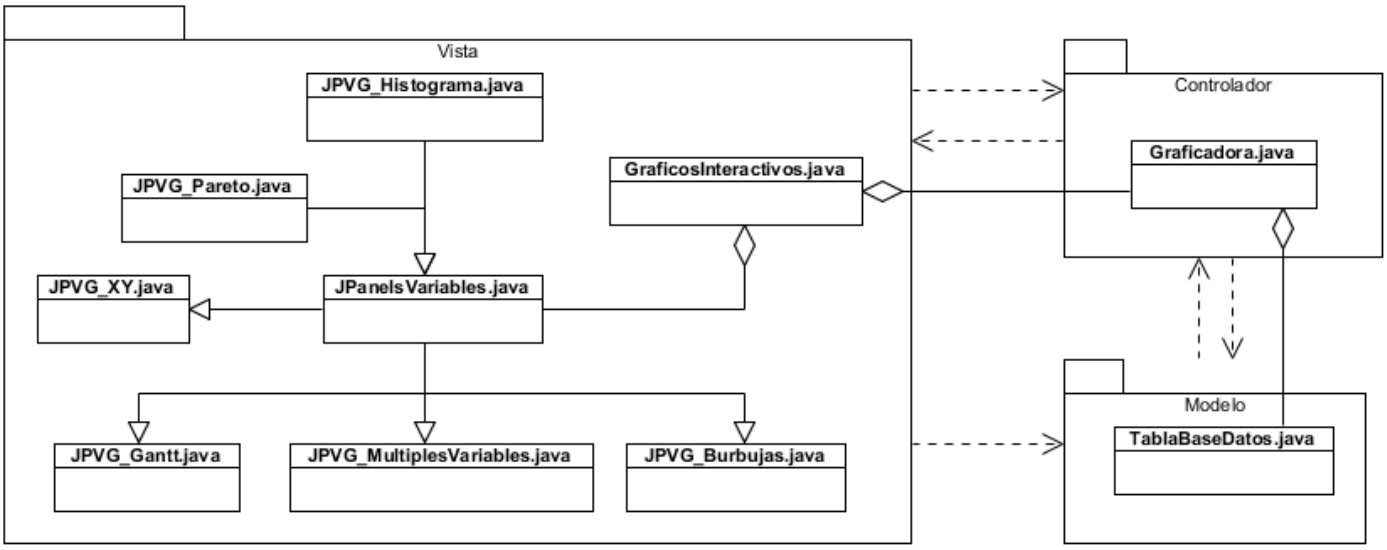

Fig. 2 DCD Generar gráficos

2.6.2 Patrones de diseño

Los patrones de diseño constituyen el otro nivel de abstracción de la arquitectura de *software*, cuyo objetivo es precisar con detalle los subsistemas y componentes de la aplicación. Estos intervienen en la forma en la que cada capa del patrón arquitectónico seleccionado “diseña” su estructura.

Entre los patrones de diseño se encuentran los GRASP y los GOF. Según *Craig Larman* "…GRASP es un acrónimo que significa *General Responsibility Asignment Software Patterns* (patrones generales de *software* para asignar responsabilidades), nombre que se eligió para indicar la importancia de captar *(grasping)* estos principios si se quiere diseñar eficazmente el *software* orientado a objetos." (21) Dentro de la clasificación de los GRASP se encuentran:

I. Experto: Consiste en asignar una responsabilidad al experto en información, es decir, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Este patrón se evidencia a través de la clase **GraficosInteractivos.java** pues es la que cuenta con la información necesaria para crear los gráficos.

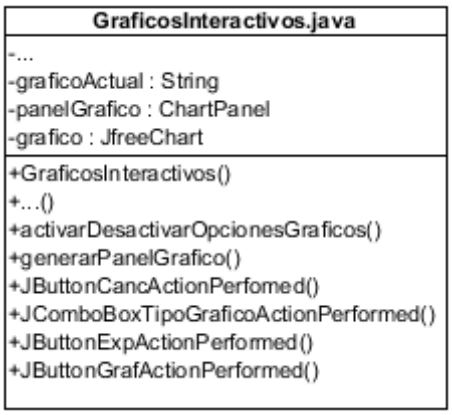

Fig. 3 Empleo del patrón experto

II. Creador: Este patrón indica a qué clase se le asigna la responsabilidad de la creación de instancias, puesto que esta posee la información necesaria para la creación de este objeto. La clase **GraficoInteractivos.java** necesita crear una instancia de la clase **Graficadora.java** para realizar sus funciones.

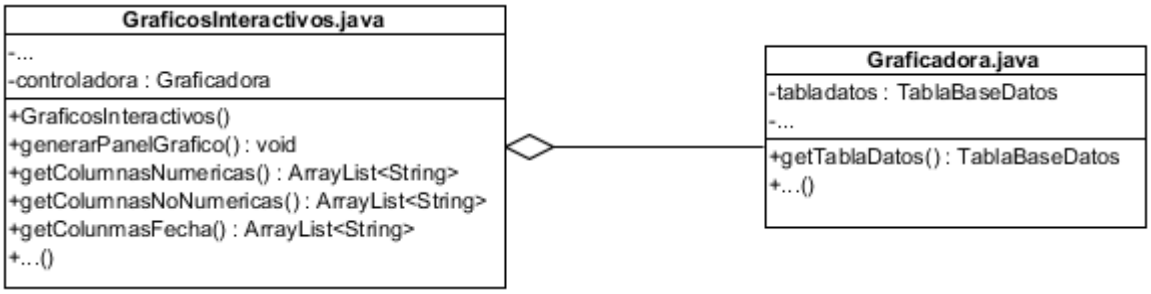

Fig. 4 Ejemplo del patrón creador

III. Bajo acoplamiento: Con este patrón se persigue lograr poca dependencia entre las clases, evitando así que cualquier cambio que pueda suceder durante el desarrollo del módulo no sea de gran repercusión. La clase **Graficadora.java** solo depende de la clase **TablaBaseDatos.java** para realizar sus funciones.

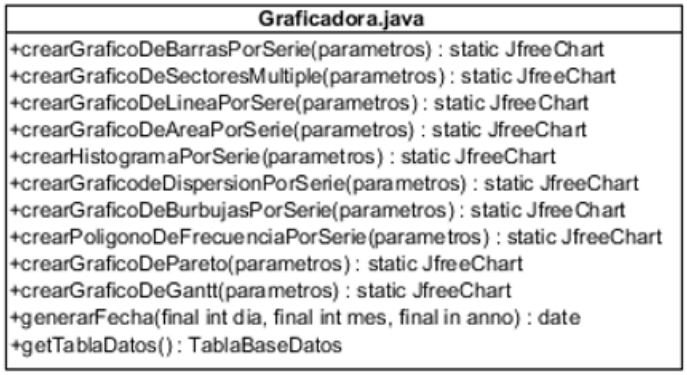

Fig. 5 Ejemplo de los patrones bajo acoplamiento y alta cohesión

IV. Alta Cohesión: Cada clase realiza una labor única dentro de un sistema. Los métodos que satisfacen las necesidades de la aplicación se encuentran implementados en diferentes clases, librándose así de la sobrecarga de responsabilidades de alguna clase en específico. La clase **Graficadora.java** posee solo los métodos necesarios para generar los gráficos, sin sobrecargarla con otros métodos que no tengan que ver directamente con ella.

V. Controlador: Este patrón le asigna la responsabilidad de un manejador de eventos de un sistema a una clase que represente un sistema global. Este patrón se encarga de que una clase actúe como intermediaria para el manejo de eventos. La clase **Graficadora.java** es la encargada de manejar los eventos de la clase **GraficosInteractivos.java**

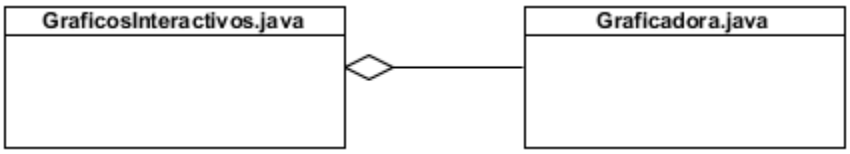

Fig. 6 Ejemplo del patrón controlador

En cuanto a los patrones GOF, sus siglas responden a *Gang of Four* o banda de los 4, compuesta por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, autores del libro "*Design Patterns—Elements of Reusable software*". Estos patrones se clasifican en tres grandes grupos:

- ✓ Patrones estructurales: describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades.
- ✓ Patrones creacionales: Corresponden a patrones de diseño *software* que solucionan problemas de creación de instancias, también ayudan a encapsular y abstraer dicha creación.
- ✓ Patrones de comportamiento: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Para la implementación de las interfaces de los gráficos se hizo uso del patrón creacional Fábrica abstracta, que es muy empleado para crear diferentes familias de objetos, evidenciándose en la implementación de los métodos de la clase **Graficadora.java** y se ve como resultado en los componentes de la ventana principal del módulo, los botones, el menú y entre otros.

Conclusiones parciales del capítulo

Partiendo de la propuesta de solución con vista a la implementación del módulo se identificaron 12 Historias de Usuario donde cinco de ellas son de prioridad muy alta, seis de prioridad alta y una de prioridad media. Se definieron 22 requisitos funcionales y cinco requisitos no funcionales para la realización del mismo. Se decidió que la implementación se realizara en 3 iteraciones, con un total de 11.3 semanas de duración. Se elaboraron dos tarjetas CRC y a partir de éstas se realizó un diagrama de clases del diseño, en el que se evidenció la utilización del patrón de arquitectura Modelo-Vista-Controlador, así como el empleo de los patrones de diseño experto, creador, alta cohesión, bajo acoplamiento, controlador y el patrón de diseño creacional fábrica abstracta.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO

Introducción

De acuerdo a la metodología propuesta, la fase de codificación durante el proceso de desarrollo del *software* es importante, debido a que en ella se desarrolla cada funcionalidad del sistema y se comprueba que se realizó exactamente lo que estaba propuesto. En el presente capítulo se muestran los estándares de codificación aplicados y se describen las pruebas realizadas al módulo, con el objetivo de comprobar el cumplimiento de las tareas de ingeniería planificadas para su desarrollo.

3.1 Codificación

En la fase de planificación, se definieron las funcionalidades necesarias para el desarrollo del módulo a partir de la realización de las HU. En esta fase de codificación esas funcionalidades se dividen en tareas más pequeñas, que se conocen como Tareas de Ingeniería (TI). Las TI son las distintas funcionalidades operativas que conforman una HU y que permiten comprobar si se está trabajando bien o no. (33) Este trabajo conjunto constituye un paso decisivo para comenzar la implementación del *software* y permite organizar el trabajo en pasos lógicos, de acuerdo a lo planificado en esa HU. A continuación se muestran las TI del módulo propuesto:

Tabla 9. TI Importar fichero de datos

Tarea de Ingeniería	
Número de la tarea: 1	**Número de la HU:** 1
Nombre de la tarea: Importar fichero de datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha de inicio: 24/1/2015	**Fecha fin**: 31/1/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para seleccionar la fuente de datos se llama el método *construirFichero (String tipo, String entidad, String bd, String ruta)* de la clase **Graficadora.java** este método va a permitir obtener la información para representarla en los gráficos implementados.	

Observaciones

Tabla 10. TI Crear gráfico de barras

Tarea de Ingeniería	
Número de la tarea: 2	**Número de la HU:** 2
Nombre de la tarea: Generar gráfico de barras	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 1/2/2015	**Fecha de inicio**: 8/2/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico de barras se llama el método *crearGraficoDeBarrasPorSerie* (*ArrayList<String> nombresColumnas,ArrayList<LinkedList<String>> columnasDeValores, LinkedList<String> columnaDeSerie,String titulo, String etiquetaEjeY, String etiquetaEjeX, boolean vertical, boolean en3D, boolean leyenda, boolean tooltips)* de la clase **Graficadora.java**.	
Observaciones	

Tabla 11. TI Exportar gráfico a PDF

Tarea de Ingeniería	
Número de la tarea: 12	**Número de la HU:** 12
Nombre de la tarea: Exportar gráfico a PDF	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha de inicio: 27/4/2015	**Fecha de inicio**: 4/5/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	

Descripción: Para exportar un gráfico a PDF se llaman a los métodos de la clase **ExportarPDF.java** que permite exportar los gráficos mostrados en el módulo**.**
Observaciones

Para ver el resto de las TI **ver Anexo 2: Tareas de ingeniería**.

3.2 Estándares de codificación

Los estándares de codificación son normas que se siguen para escribir armónicamente el código fuente de un proyecto, que deben cumplir con cada uno de sus aspectos para asegurar que cada programador del proyecto trabaje de manera organizada y coordinada. (34) El Sistema Auditoría de Datos propone una serie de convenciones o estándares de códigos enfocados a la estructura, apariencia física del sistema, comprensión y mantenimiento del código. A continuación se muestran algunas de estas convenciones tenidas en cuenta para la implementación del módulo de Gráficos v2.0:

- ✓ **Identación**

Se deben emplear al menos cuatro espacios como unidad de identación.

```
public static JFreeChart crearGraficoDeSectoresMutiple(ArrayList<String> nombresColumnas,
```

Fig. 7 Ejemplo de identación

- ✓ **Comentarios**

Existen dos tipos de comentario, los de implementación y los de documentación, estos últimos se realizan de manera automática solo en Java y se encuentran limitados por /**…*/.

- ✓ **Declaraciones**

Se realizará una declaración por línea pues facilita los comentarios también. Al declarar clases e interfaces se siguieron las siguientes reglas de formato:

I. No se realiza ningún espacio entre el nombre del método y el paréntesis “(” que abre su lista de parámetros.

   La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.

II. La llave de cierre "}" empieza una nueva línea identada para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{".

III. Los métodos se separaron con una línea en blanco.

```
public static JFreeChart crearGraficoDeDispersion(LinkedList<String> columnaEjeX,
        LinkedList<String> columnaEjeY, String titulo, String etiquetaEjeY,
        String etiquetaEjeX, boolean vertical, boolean leyenda, boolean tooltips, boolean url) throws Exception {

    final XYSeriesCollection juegoDeDatos = crearXYSeriesCollection(columnaEjeX, columnaEjeY);

    final JFreeChart grafico = ChartFactory.createScatterPlot(titulo, etiquetaEjeX,
            etiquetaEjeY, juegoDeDatos, vertical ? PlotOrientation.VERTICAL
            : PlotOrientation.HORIZONTAL, leyenda, tooltips, url);
    mejorarGraficoXYPlot(grafico);
    return grafico;
}
```

Fig. 8 Ejemplo de declaraciones

Inicialización

Se inicializarán las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

Sentencias del paquete

La primera línea no comentada de un fichero fuente debe ser la sentencia de paquete, que indica el paquete al que pertenecen las clases incluidas en el fichero fuente.

```
package Vistas.GraficosInteractivos;
```

Fig. 9 Ejemplo de sentencias de paquete

Sentencias if

Las sentencias if-else debe tener la siguiente forma y deben usar siempre llaves {}.

```
if (llaveCompararPorColumna != null && !llaveCompararPorColumna.equals("")) {
    try {
        valor = Double.parseDouble(llaveCompararPorColumna);
    } catch (NumberFormatException e) {
        valor = 0;
    }
} else {
    valor = 0;
}
llaveCompararPorFila = columnaDeSerie.get(j);
llaveCompararPorColumna = nombresColumnas.get(i);
juegoDDatos.addValue(valor, llaveCompararPorFila, llaveCompararPorColumna);
```

Fig. 10 Ejemplo de sentencias *if*

Sentencias *for*

Debe declararse de la siguiente forma y si contiene un ***if*** deberá declararse dentro del ***for*** como a continuación se muestra:

```
for (int i = 0; i < columnasDeValores.size(); i++) {
    valoresGraficar = columnasDeValores.get(i);
    for (int j = 0; j < valoresGraficar.size(); j++) {
        llaveCompararPorColumna = valoresGraficar.get(j);//reusando variable
        if (llaveCompararPorColumna != null && !llaveCompararPorColumna.equals("")) {
            try {
                valor = Double.parseDouble(llaveCompararPorColumna);
            } catch (NumberFormatException e) {
                valor = 0;
            }
        } else {
            valor = 0;
        }
        llaveCompararPorFila = columnaDeSerie.get(j);
        llaveCompararPorColumna = nombresColumnas.get(i);
        juegoDDatos.addValue(valor, llaveCompararPorFila, llaveCompararPorColumna);
    }
}
```

Fig. 11 Ejemplos de sentencias *for* anidado

Variables

Todas las instancias y variables de clase o método empezarán con minúscula y al final de la declaración se colocará punto y coma. Las palabras internas que la forman empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres sub-guión "_" ni signo del dólar.

```
valor = 0;
```

Fig. 12 Ejemplo de declaración de variables

Interfaz principal del Módulo de Gráficos v2.0

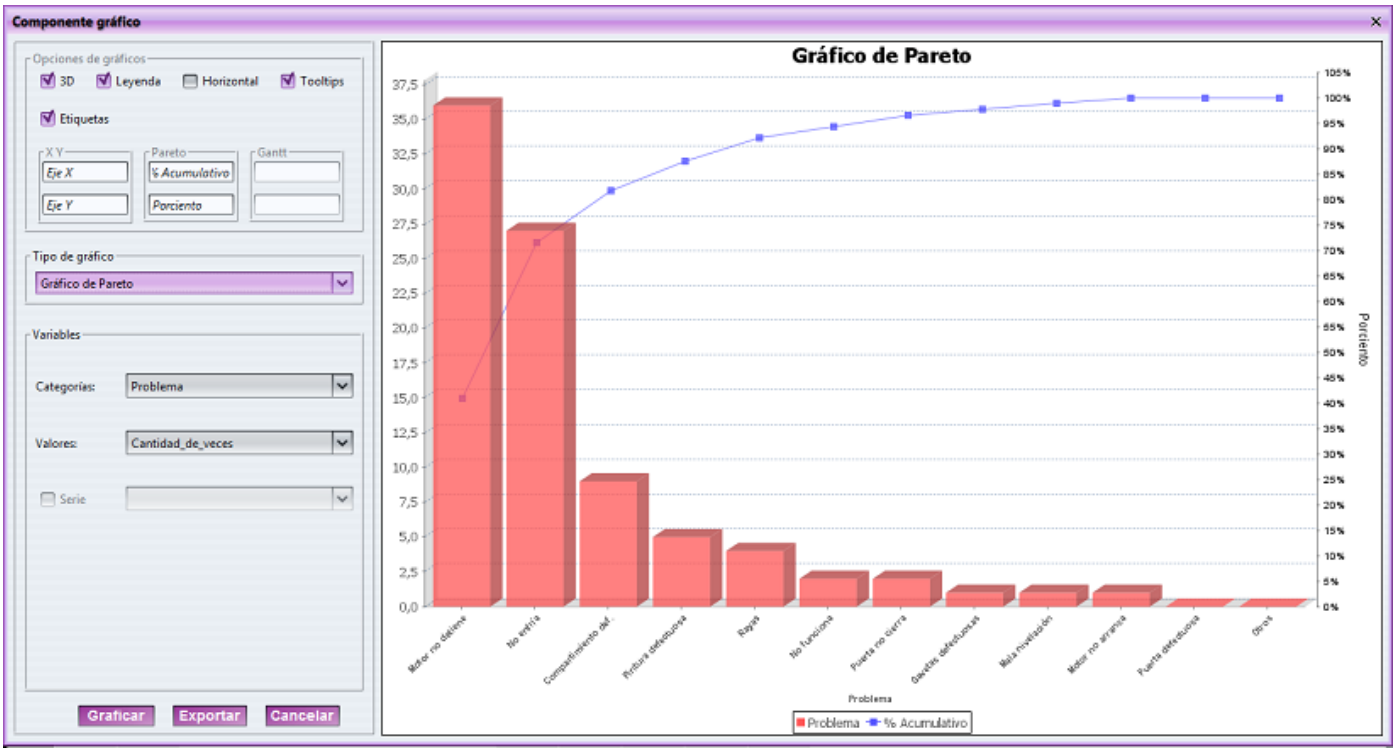

Fig. 13 Interfaz principal del módulo implementado

3.3 Realización de pruebas al módulo

Uno de los pilares fundamentales en el proceso de desarrollo de un *software* es la realización de las pruebas porque se reduce el número de errores no detectados, así como el tiempo entre la introducción de éste y su detección dentro del sistema. El uso de las pruebas es la manera de comprobar que las funcionalidades que se van desarrollando funcionan correctamente y cumplen con lo requerido por el cliente. En el presente trabajo se realizaron las siguientes pruebas:

- ✓ Pruebas unitarias para comprobar cada módulo del sistema buscando errores en el funcionamiento de éstos como sistema independiente.
- ✓ Pruebas de integración para comprobar que el sistema y sus partes funcionan correctamente como un todo.
- ✓ Pruebas funcionales para comprobar que las funcionalidades previamente diseñadas para el *software* se ejecutan correctamente.

- ✓ Pruebas de aceptación para comprobar que los requisitos implementados satisfacen las necesidades del cliente.

3.3.1 Pruebas unitarias

Las pruebas unitarias son una forma de probar el buen funcionamiento de un módulo o una parte del sistema, con el fin de asegurar el correcto funcionamiento de todos los módulos por separados. Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas comprueban la forma en que las funciones y los métodos trabajan en cada caso particular. (35)

Estas pruebas también son llamadas pruebas modulares ya que permiten determinar si un módulo del programa está listo y correctamente terminado. El objetivo fundamental de las pruebas unitarias es asegurar el correcto funcionamiento de las interfaces o flujo de datos entre componentes. Teniendo en cuenta las características de las pruebas unitarias se decide utilizar el método de caja blanca para garantizar que se ejerciten por lo menos una vez todos los caminos independientes del módulo implementado.

Método de prueba aplicado

Este método de prueba está centrado en la estructura de control del programa. Permite examinar el estado del programa en varios puntos para determinar si el estado real coincide con esperado. Una de las técnicas más empleadas por este método para comprobar el funcionamiento correcto de las estructuras de los datos es la técnica del camino básico.

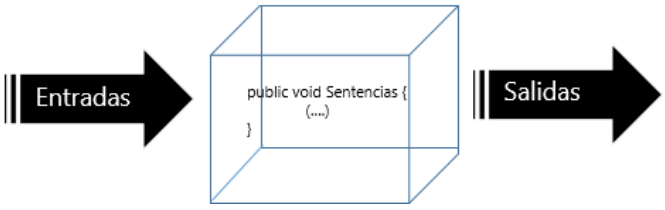

Fig. 14 Representación del método de caja blanca

- ✓ Camino Básico

Esta técnica permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez.

```
public static JFreeChart crearHistograma(LinkedList<String> columnaDeValores,
        String titulo, String etiquetaEjeY, String etiquetaEjeX,
        boolean vertical, boolean leyenda, boolean tooltips, boolean url) throws Exception {

    final XYSeries serie = new XYSeries("");
    int cont = 0;
    double valor;
    String string;
    for (int i = 0; i < columnaDeValores.size(); i++) {
        string = columnaDeValores.get(i);
        if (string != null && !string.equals("")) {
            try {
                valor = Double.parseDouble(columnaDeValores.get(i));
            catch (NumberFormatException e) {
                valor = 0;
            }
        } else {
            valor = 0;
        }
        serie.add(++cont, valor);
    }
    final XYSeriesCollection coleccionDSeries = new XYSeriesCollection(serie);
    final JFreeChart grafico = ChartFactory.createHistogram(titulo, etiquetaEjeX,
            etiquetaEjeY, coleccionDSeries, (vertical ? PlotOrientation.VERTICAL : PlotOrientation.HORIZONTAL),
            leyenda, tooltips, url);

    final XYPlot plot = grafico.getXYPlot();
    final XYItemRenderer renderer = plot.getRenderer();
    final GradientPaint gradientGray = new GradientPaint(0.0F, 0.0F, new Color(142, 180, 227), 0.0F, 0.0F, new Color(208, 208, 208));
    final GradientPaint gradientRed = new GradientPaint(0.0F, 0.0F, new Color(55, 96, 146), 0.0F, 0.0F, new Color(3, 8, 111));
    renderer.setSeriesPaint(0, gradientGray);
    renderer.setSeriesPaint(1, gradientRed);
    mejorarGraficoXYPlot(grafico);
    return grafico;
}
```

Fig. 15 Ejemplo de algoritmo

Para realizar la prueba es necesario realizar primeramente el análisis de la complejidad del algoritmo sobre el que se va a realizar la prueba, con el fin de calcular los valores de la complejidad ciclomática. En la figura se representan los distintos componentes del grafo de flujo asociado que se genera como es el caso de:

- ✓ Los nodos que son círculos representados en el grafo de flujo que representan una o más secuencias del procedimiento, donde cada nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al principio y al final del grafo.
- ✓ Los nodos predicados que son los que contienen una condición y se caracterizan porque de ellos salen una o más aristas.
- ✓ Las aristas son las flechas del grafo, iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando este no represente la sentencia de un procedimiento.
- ✓ Las regiones son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo como una región más. Las regiones se enumeran: siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico del procedimiento.

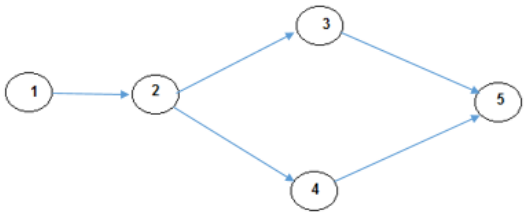

Fig. 16 Grafo de flujo asociado al algoritmo Crear Histograma.

La complejidad ciclomática coincide con el número de regiones del grafo de flujo y se define como:
V (G) = Aristas – Nodos + 2, o también como **V (G) = Nodos predicados + 1**. A partir del grafo anterior la complejidad ciclomática sería la siguiente:
Como el grafo tiene dos regiones, V (G) = 2
Como tiene 5 aristas y 5 nodos, V (G) = 5 – 5 + 2 = 2
Como tiene un solo nodo predicado, V (G) = 1 + 1 = 2
Teniendo la complejidad ciclomática se obtiene el número de caminos independientes que dan un valor límite para el número de pruebas que se le debe realizar al módulo,1-2-3-5 y 1-2-4-5 así para cada camino se realiza un caso de prueba.

Caso de prueba para el camino básico no.1
Descripción: Los datos de entrada deben cumplir con los siguientes requisitos:
Una lista para almacenar los valores de la columna asociada a la categoría que se desea visualizar, una variable inicializada en cero, una variable que almacene datos de tipo cadena y otra variable que almacene números de todo tipo.
Condición de ejecución: Una lista vacía, una variable de tipo *int* inicializada en cero, una variable de tipo *String* y otra variable de tipo *double*.
Resultados esperados: Se espera que el sistema muestre una excepción ya que no existirán variables cuantitativas para mostrar el gráfico.
Seguidamente se muestra un ejemplo de manera visual de este caso de prueba:

	(Inicio)Fech...	(Inicio)Fech...	(Fin)Fecha_...	(Fin)Fecha_...	Tareas
1	2015/01/12	2015/01/14	2015/01/16	2015/01/18	Analysis
2	2015/01/17	2015/01/19	2015/01/18	2015/01/22	Design
3	2015/01/18	2015/01/18	2015/01/23	2015/01/25	Implement...
4	2015/01/19	2015/01/19	2015/01/24	2015/01/27	Testing
5	2015/01/20	2015/01/23	2015/01/21	2015/01/24	Deployment
6	2015/01/22	2015/01/24	2015/01/23	2015/01/25	Assistment
7	2015/01/25	2015/01/25	2015/01/28	2015/01/30	Support
8	2015/01/27	2015/01/28	2015/01/30	2015/02/03	Maintenance

Fig. 17 Ejemplo de variable tipo *String*

El algoritmo lee la variable como cadena porque está escrita en formato de fecha por lo que ya no sería una variable cuantitativa sino cualitativa, en este caso que el sistema debe lanzar un mensaje de error como se muestra en la figura 27.

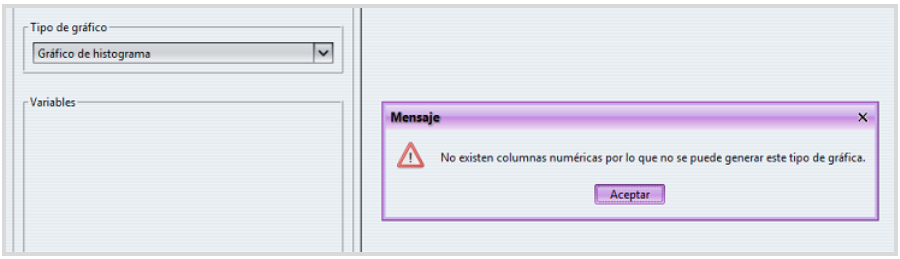

Fig. 18 Resultado esperado para las variables de tipo *String*

En caso de que la tabla insertada contenga valores numéricos y no contenga en ningún otro campo de la lista una serie mediante la cual se pueda comparar cada elemento de la lista (dígase meses, años, nombres u otros elementos comparativos), se generará el gráfico de histograma, pero sin especificar la serie lanzando otro mensaje de error para comunicárselo al usuario final.

	Edad	PESO	Trabajo	Esfuerzo
1	10	15	4	0
2	15	20	3	0
3	20	25	6	0
4	25	30	4	0
5	30	35	15	0
6	35	40	14	0
7	40	45	4	0
8	45	50	16	0
9	50	55	13	0
10	55	60	14	0
11	60	65	17	0
12	65	70	16	0
13	70	75	3	0
14	75	80	15	0
15	80	85	3	0

Fig. 19 Ejemplo de variable de tipo *String* vacía

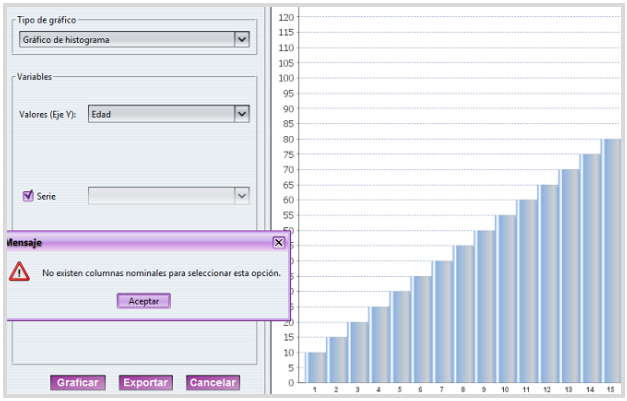

Fig. 20 Resultado esperado de variable tipo *String* vacía

3.3.2 Pruebas de integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras, al mismo tiempo, se lleva a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados en unidad y estructurar un programa que esté de acuerdo con el que dicta el diseño. (13)

Durante la realización de las pruebas de integración en el Módulo de Gráficos v2.0, al incorporar las clases de la nueva versión del módulo en el sistema informático AUDAT se detectó 1 NC de tipo significativa. La que fue resuelta satisfactoriamente en una segunda iteración al declarar un objeto de tipo **GraficosInteractivos.java** en la clase **AUDATPrincipal.java.**

3.3.3 Pruebas funcionales

Se realizan con el objetivo de comprobar que los sistemas desarrollados funcionan de acuerdo a las especificaciones descritas, definidas por el cliente, lo que posibilita detectar los posibles defectos derivados de errores en la fase de codificación. Teniendo en cuenta las características del módulo se decide aplicar el método de caja negra para comprobar que luego de ejecutada cada una de las funcionalidades asociadas a los gráficos de módulo se muestre correctamente el gráfico correspondiente.

Método de Caja Negra

Como se explicó anteriormente, debido a las características de la herramienta se decide utilizar el método de caja negra ya que su objetivo fundamental es la interacción con el *software*, entendiendo qué es lo que hace, sin dar importancia a cómo lo hace. Este método se enfoca a las necesidades del cliente, lo involucra como un miembro más del equipo, cumpliendo con las características que propone la

metodología XP. Para su realización debe estar muy bien definida la interfaz del módulo. Con este tipo de prueba se intenta detectar funciones incorrectas o ausentes, errores de interfaz e incluso problemas de rendimiento.

Fig. 21 Representación del método de caja negra

- Partición de equivalencia

Se considera una de las técnicas más efectivas dentro de este método pues permite examinar los valores válidos e inválidos de las entradas existentes en el *software*. Para eso se identifican las clases de equivalencia. Se toma cada condición de entrada o salida (usualmente frase o sentencia en la especificación) y se la divide en 1 o más grupos. Una cierta combinación de clases de entrada da como resultado una combinación de clases de salida.

Caso de prueba

Los casos de pruebas son un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular. Comprueban si el producto satisface los requisitos del usuario final, tal y como se describen en las HU, además de comprobar si el producto se comporta tal y como se describen en las especificaciones funcionales del diseño. A continuación se muestra un ejemplo de los casos de pruebas realizados al módulo, para comprobar que la funcionalidades descritas en la HU “Generar gráfico de *Gantt*” funcionaba correctamente, para verificar su funcionamiento se definieron dos variables que fueron evaluadas en dos escenarios diferentes.

Tabla 12. Clasificación de las variables

No	Nombre del campo	Clasificación	Valor nulo	Descripción
1	3D	Campo de selección	no	Se muestra el gráfico en 3D
2	Horizontal	Campo de selección	no	Se muestra el gráfico horizontal
3	Leyenda	Campo de selección	no	Se muestra el gráfico con la leyenda

4	*Tooltips*	Campo de selección	no	Se muestra los valores del gráfico cuando se le pasa el *mouse* por encima
5	Etiquetas	Campo de selección	no	Se muestra el gráfico con las etiquetas
6	Seleccionar tipo de gráfico	Lista desplegable	no	Se selecciona le tipo de gráfico a generar.
7	Variables	Campo de texto	no	Se muestran las columnas con los datos

Tabla 13. Caso de prueba: Generar gráfico de barras

Escenario	Descripción	V1	V2	V3	V5	V6	V7	Respuesta del sistema	Flujo central
EC 1.1 Generar gráfico de barras	Se selecciona en el espacio "Tipo de Gráfico" la opción Gráfico de barras	V	V	V	V	V	V	El sistema muestra el gráfico de barras en 3D, horizontal, con la leyenda, con sus etiquetas y con sus *tooltips*.	-Seleccionar la fuente de datos. -Seleccionar el ícono de gráfico. -Seleccionar en la opción "Tipo de gráfico" el gráfico de barras. -Se oprime el botón "Graficar"
EC 1.1 Generar gráfico de barras con datos vacíos	Se selecciona en el espacio "Tipo de Gráfico" la opción Gráfico de barras	V	V	V	V	V	I	El sistema muestra un mensaje de error: "No se puede mostrar esta gráfica porque no existen datos en el sistema".	-Seleccionar la fuente de datos. -Seleccionar el ícono de gráfico. -Seleccionar en la opción "Tipo de gráfico" el gráfico de barras. -Se oprime el botón "Graficar". -Se muestra el mensaje

									de error.

Resultados de las pruebas funcionales

La aplicación de las pruebas funcionales permitió detectar un total de 4 NC (No Conformidades) clasificadas como Significativas. En una primera iteración se detectaron 3 NC, en la segunda iteración se detectó 1 NC, siendo resueltas satisfactoriamente lo que permitió que en una tercera iteración no se detectaran NC. En la Figura 22 se presentan los resultados de estas pruebas.

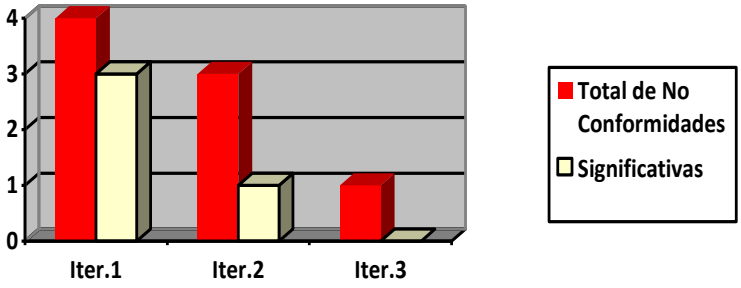

Fig. 22 Resultados de las pruebas funcionales

3.3.4 Pruebas de aceptación

Las pruebas de aceptación se realizan sobre el producto terminado. Están concebidas para que sea un usuario final quien detecte los posibles errores. En esta prueba se evalúa el grado de calidad del *software* con relación a todos los aspectos relevantes para que el uso del producto se justifique. Este tipo de pruebas son comúnmente realizadas por el usuario final, quien debe informar de todas las deficiencias o errores que encuentre antes de dar por aprobado el sistema definitivamente. (36)

Para asegurarse que el módulo desarrollado cumple con sus requisitos, se decidió realizarle pruebas de aceptación de tipo alfa. En este tipo de prueba se utiliza el *software* de forma natural con el desarrollador como observador del cliente en un ambiente controlado. El Módulo de Gráficos v2.0 fue probado por el especialista del proyecto AUDAT el cual certificó mediante la carta de aceptación que el sistema cumple con los requisitos establecidos por el cliente. **(Ver anexo 5)**

Conclusiones parciales del capítulo

Durante el presente capítulo se implementaron las 12 HU definidas en el capítulo anterior, las cuales quedaron reflejadas en 12 tareas de ingeniería. Se definió el estándar de codificación a utilizar para la implementación del módulo, garantizando la uniformidad y organización del código. Clientes y desarrolladores le realizaron pruebas al módulo utilizando los métodos de caja negra y caja blanca a través de las técnicas de partición de equivalencia y camino básico respectivamente, garantizando el correcto funcionamiento de las entradas y salidas de la aplicación, tanto en las interfaces como en la ejecución del código. Se realizaron pruebas de integración y se resolvieron en dos iteraciones todas las no conformidades detectadas. Se llevaron a cabo las pruebas de aceptación realizadas por el cliente, para obtener un producto con la calidad requerida.

CONCLUSIONES GENERALES

Al finalizar la presente investigación se arribaron a las siguientes conclusiones:

- ✓ El estudio realizado sobre las características de los gráficos contribuyó a la selección de las tecnologías y las herramientas adecuadas para desarrollar el Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos, así como de la selección de XP como metodología para guiar todo el proceso de desarrollo del mismo.
- ✓ Se elaboraron los artefactos de acuerdo a la metodología propuesta que permitieron establecer las pautas para el desarrollo del Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos.
- ✓ Se llevó a cabo un módulo capaz de generar gráficos que le permitirá a los especialistas de la Contraloría General de República de Cuba analizar la información auditada.
- ✓ Las pruebas realizadas al módulo durante su etapa de desarrollo demostraron que se cumplió satisfactoriamente con los requisitos propuestos, asegurando de esta manera la calidad del producto final.

RECOMENDACIONES

- ✓ Se recomienda agregar otros gráficos en el Módulo de Gráficos v2.0 para el Sistema Auditoría de Datos que permitan a los especialistas de la Contraloría General de la República nuevas interpretaciones de la información auditada.
- ✓ Se recomienda que el módulo implementado permita exportar los gráficos en otros formatos.

REFERENCIAS BIBLIOGRÁFICAS

1. **Calvete, Marta Graño. 2008.** Libros de Cabecera. [En línea] 2008 [Citado el: 24 de septiembre de 2014.] https://www.librosdecabecera.com/articulos/que-debo-preparar-antes-de-la-primera-auditoria-en-mi-empresa.

2. **Rivas, Catalina. 2014.** La auditoría en el concepto actual. [En línea] 2014. [Citado el: 24 de septiembre de 2014.] http://www.gestiopolis.com/recursos/documentos/fulldocs/fin/auditcontxactual.htm.

3. **Echenique, Jose Antonio. 2015**. Resumen Ejecutivo de la Auditoría Informática. [En línea] 2015 [Citado el: 30 de septiembre de 2014.] http://www.buenastareas.com/ensayos/Resumen-Ejecutivo-De-La-Auditoria-Inform%C3%A1tica/7028692.html.

4. **Castello, Ricardo.** Auditoría en entornos informáticos. [En línea] [Citado el: 2 de octubre de 2014.] www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&ved=0CE4QFjAIahUKEwjGuMXV04vGAhVERIwKHUWOAOs&url=http%3A%2F%2Fecon.unicen.edu.ar%2Fmonitorit%2Findex.php%3Foption%3Dcom_docman%26task%3Ddoc_download%26gid%3D552%26Itemid%3D19&ei=e5l7VYaPEcSIsQTFnILYDg&usg=AFQjCNGyX41tx9VcBI4MDr9tf2BnV3D3QQ&bvm=bv.95515949,d.cWc&cad=rja.

5. **Fisher, Royal P.** Seguridad en los Sistemas Informáticos. [En línea] [Citado el: 2 de octubre de 2014.] http://books.google.com.cu/books?id=_Hu6Zu6VLP4C&pg=PA25&lpg=PA25&dq=componentes+auditables&source=bl&ots=zQiI5Q0B72&sig=ndGSk6wblVx1jCBfr1_GY0NfoXc&hl=es-419&sa=X&ei=FZRWVKraHMibNq-_gOgJ&sqi=2&ved=0CEoQ6AEwCA#v=onepage&q=componentes%20auditables&f=false.

6. **Perez, Jessica. 2011.** Las bases de datos, su seguridad y auditoría. El caso de MySQL. [En línea] 2011. [Citado el: 3 de octubre de 2014.] http://e-archivo.uc3m.es/bitstream/handle/10016/12011/PFC_Jessica_Perez_Sandoval.pdf?sequence=1. Bloque III, Capítulo 4, epígrafe 4.3, pág. 137

7. **Alvarado, Jorge Andrés. 2008.** Fundamentos de inferencia estadística. [En línea] 2008. [Citado el: 3 de octubre de 2014.] https://books.google.com.cu/books?hl=es&lr=&id=3uhUqvF0_84C&oi=fnd&pg=PA13&dq=alvarado+inferencia&ots=DD5El8w5Op&sig=778OW0Ld6P74AquYASNYjWvoWrQ#v=onepage&q=alvarado%20inferencia&f=false Capítulo 1, epígafe pág. 17

8. **Pértega Díaz S., Pita Fernández S. 2001.** Representación gráfica en el Análisis de Datos. [En línea] 2001 [Citado el: 9 de octubre de 2014.] www.fisterra.com/mbe/investiga/graficos/graficos.pdf. epígrafe Análisis Descriptivo pág.1

9. **Serrano, Luis. 2009.** Tendencias actuales de la educación estocástica. [En línea] 2009 [Citado el: 6 de octubre de 2014.] http://www.pucrs.br/famat/viali/graduacao/matematica/material/referencias/libroluis.pdf#page=133. Capítulo 7 pág. 136

10. **Orellana, Liliana. 2001.** Estadística Descriptiva. [En línea] 2001 [Citado el: 6 de octubre de 2014.] http://www.hacienda.go.cr/cifh/sidovih/cursos/material_de_apoyo-f-c-cifh/1materialdeapoyocursoscifh/4estad%C3%ADsticabasica/estadisticadescriptiva-lillianaorellana.pdf Capítulo 3 pág. 20-28, Capítulo 5 pág. 51-61

11. **Rea, Karina. 2010.** Análisis de la problemática en el sistema de gestión de calidad de la agencia aduanal grupo Eduardo Díaz S. C., y propuesta de mejora para el cumplimiento de la norma ISO 9001:2008**.** [En línea] 2010. [Citado el: 4 de noviembre de 2014.] http://tesis.bnct.ipn.mx/bitstream/handle/123456789/5975/A2.746.pdf?sequence=1

12. **Hernández Silvia M. 2010.** Excel Básico [En línea] 2010. [Citado el: 12 de noviembre de 2014] http://repository.uaeh.edu.mx/bitstream/bitstream/handle/123456789/11330/LECT25.pdf?sequence=1. Epígrafe 10, pág. 7

13. **Pressman, Roger. 1997** Ingeniería del software: un enforque práctico. 1997 [En línea] https://books.google.com.cu/books?hl=es&lr=&id=8UV5jxkuBZIC&oi=fnd&pg=PP3&dq=metodologia+de+desarrollo+de+software+pressman&ots=wJQr0NOlFN&sig=pFrPcpP-B33dFFz04h9gAzfZiYM#v=onepage&q=metodologia%20de%20desarrollo%20de%20software%20pressman&f=false pág. 57

14. **Portal Oficial de XP.** [En línea] 2014. [Citado el: 16 de octubre de 2014.] www.extremeprogramming.org.

15. **Portal Oficial de HighCharts.** [En línea] 2014. [Citado el: 20 de octubre de 2014] www.highcharts.com

16. **Portal Oficial de FusionChart.** [En línea] 2014. [Citado el: 20 de octubre de 2014] http://www.fusioncharts.com/

17. **Portal Oficial de JfreeChart.** [En línea] 2014. [Citado el: 20 de octubre de 2014.] www.jfree.org/jfreechart.

18. **Portal Oficial de Netbeans.** [En línea] 2014. [Citado el: 20 de octubre de 2014] https://netbeans.org/community/releases/80/

19. **López G., Amparo.** Introduccion Al Desarrollo De Programas Con Java [En línea][Citado el: 2 de noviembre de 2014.] https://books.google.com.cu/books?id=29zE8HTdJ1QC&pg=PA8&lpg=PA8&dq=bibliotecas+graficas+java+j&source=bl&ots=MV6ZJ_OKW0&sig=mD5jk2YVjTY91QeYB1sZgJR1vsk&hl=es&sa=X&ved=0CDUQ6AEwA2oVChMIg-jXpoeVxgIVuCWMCh3KAwAx#v=onepage&q=bibliotecas%20graficas%20java%20j&f=false Capítulo 1 pág. 8

20. **Portal Oficial de Visual Paradigm.** [En línea] 2014. [Citado el: 29 de noviembre de 2014.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.

21. **Larman, Craig. 2006** UML y Patrones. Prentice Hall. Introducción al análisis y diseño orientado a objetos. [En línea] 2006. [Citado el: 1 de febrero de 2015] https://docs.google.com/file/d/0B_MWRyBDHOk-UDhZUzNFU2V4OWs/edit?pli=1 Capítulo 9 pág. 85

22. **Joskowicz, Jose. 2008** Reglas y Prácticas en eXtreme Programming [En línea] [Citado el: 2 de marzo de 2015.] http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf epígrafe 4.1, pág. 10

23. **Fernández, Gerardo. 2002** [En línea] 2002 [Citado el: 9 de febrero de 2015.] http://www.um.edu.ar/catedras/claroline/backends/download.php?url=L01ldG9kb3NfQWdpbGVzL1Byb2dyYW1hY2lvbl9FeHRyZW1hLVhQLnBkZg%3D%3D&cidReset=true&cidReq=II0162004 epígrafe 3.2.3 pág. 7

24. **Pressman, Roger. 2010.** Ingeniería del software: Un enfoque práctico, 7ma edición, McGraw Hill International, Singapore. Capítulo 10, epígrafe 10.1, pág. 276

25. **Bascón, Ernesto. 2015.** El patrón de diseño Modelo-Vista-Controlador y su implementación en Java Swing [En línea] 2015 [Citado el: 5 de marzo de 2015.] http://www.academia.edu/5217432/El_patr%C3%B3n_de_dise%C3%B1o_Modelo-Vista-Controlador_MVC_y_su_implementaci%C3%B3n_en_Java_Swing epígrafe 2, pág. 494

26**.** Estándares de codificación**.** [En línea] [Citado el: 14 de abril de 2015.] https://docs.google.com/document/d/1rbxDFM0zsbFDNRZeM2FoXfRDbYSiSt6tCdbYPA0qdzs/edit?hl=en_US&pli=1

27. **Sommariva, Andrés. 2013** [En línea] 2013 [Citado el: 14 de abril de 2015.] http://www.microgestion.com/index.php/mg-developers/articulos/74-unit-test-part1-mock

29. **Muñoz P., Álvarez. 2010.** Gestión de Requisitos dirigidos por Pruebas de Aceptación [En línea] 2010. [Citado el: 26 de abril de 2015.] https://riunet.upv.es/bitstream/handle/10251/11307/Trabajo%20Fin%20Master.pdf?sequence=1 Capítulo 2, eígrafe 2.2.3 Acceptance Test Driven Development pág.35

ANEXOS

Anexo 1: Guía de observación

I. Guía de observación a:

Módulo de Gráficos v1.0 del Sistema Auditoría de Datos.

II. Objetivo:

Constatar las manifestaciones de los usuarios con el módulo, así como el empleo de los gráficos en el Módulo de Gráficos v1.0 del Sistema Auditoría de Datos.

III. Aspectos a observar:

Habilidades de los usuarios para interactuar con los gráficos.

Cantidad de gráficos existentes.

Dificultades más notables en la selección y generación de los gráficos.

Frecuencia con las que se utilizan los gráficos existentes.

Otros.

Anexo 2: Tipos de gráficos implementados

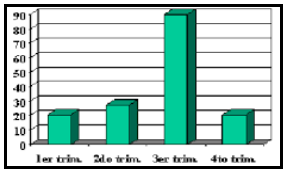

Fig. 23 Gráfico de barras

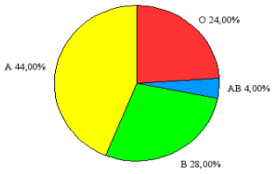

Fig. 24 Gráfico de sectores

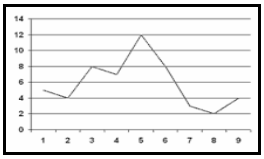

Fig. 25 Gráfico de línea

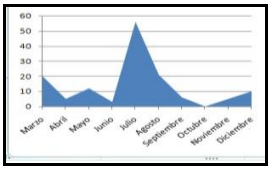

Fig. 26 Gráfico de área

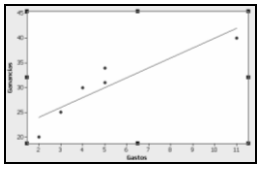

Fig. 27 Gráfico de dispersión

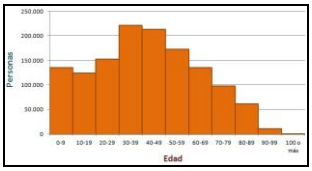

Fig. 28 Gráfico de histograma

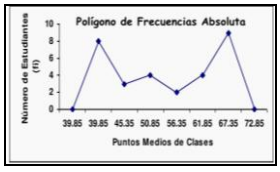

Fig. 29 Gráfico de polígono de frecuencias

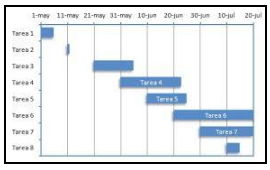

Fig. 30 Gráfico de Gantt

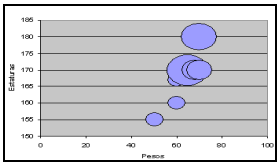

Fig. 31 Gráfico de burbujas

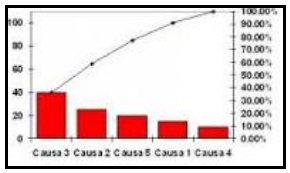

Fig. 32 Gráfico de Pareto

Anexo 3: Historias de Usuario implementadas

Tabla 14. HU Generar gráfico de sectores

Historia de Usuario	
Número: 3	**Nombre:** Crear gráfico de sectores
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada:**1
Prioridad en el negocio: Muy alta	**Puntos estimados:** 1
Riesgo de desarrollo: Alto	**Puntos reales:** 1
Descripción: Permite la creación de un gráfico de sectores a partir de la información almacenada y al mismo tiempo se pueden modificar sus características de acuerdo a la opción que se seleccione en la interfaz de la aplicación.	
Observaciones: En caso de no existir variables de acuerdo a las características de un gráfico de sectores se muestra un mensaje de error.	

Tabla 15. HU Generar gráfico de línea

Historia de Usuario	
Número: 4	**Nombre:** Crear gráfico de línea
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada:**1
Prioridad en el negocio: Muy alta	**Puntos estimados:** 1
Riesgo de desarrollo: Alto	**Puntos reales:** 1
Descripción: Permite la creación de un gráfico de línea a partir de la información almacenada y al mismo tiempo se pueden modificar sus características, de acuerdo a la opción que se seleccione en la interfaz de la aplicación.	

Observaciones: En caso de no existir variables de acuerdo a las características de un gráfico de línea se muestra un mensaje de error.

Tabla 16. HU Generar gráfico de área

Historia de Usuario	
Número: 5	**Nombre:** Crear gráfico de área
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada**:1
Prioridad en el negocio: Muy alta	**Puntos estimados:** 1
Riesgo de desarrollo: Alto	**Puntos reales:** 1
Descripción: Permite la creación de un gráfico de área a partir de la información almacenada y al mismo se tiempo pueden modificar sus características de acuerdo a la opción que se seleccione en la interfaz de la aplicación.	
Observaciones: En caso de no existir variables de acuerdo a las características de un gráfico de área se muestra un mensaje de error.	

Tabla 17. HU Generar gráfico de dispersión

Historia de Usuario	
Número: 6	**Nombre:** Crear gráfico de dispersión
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada**: 2
Prioridad en el negocio: Alta	**Puntos estimados**: 1
Riesgo de desarrollo: Alto	**Puntos reales:** 1

Descripción: Permite la creación de un gráfico de dispersión a partir de la información almacenada y al mismo se tiempo pueden modificar sus características de acuerdo a la opción que se seleccione en la interfaz de la aplicación.
Observaciones: En caso de no existir variables de acuerdo a las características de un gráfico de dispersión se muestra un mensaje de error.

Tabla 18. HU Generar gráfico de histograma

Historia de Usuario	
Número: 7	**Nombre:** Crear gráfico de histograma
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada:** 2
Prioridad en el negocio: Alta	**Puntos estimados**: 1
Riesgo de desarrollo: Alto	**Puntos reales**: 1
Descripción: Permite la creación de un gráfico de histograma a partir de la información almacenada y al mismo se tiempo pueden modificar sus características de acuerdo a la opción que se seleccione en la interfaz de la aplicación.	
Observaciones: En caso de no existir variables de acuerdo a las características de un gráfico de histograma se muestra un mensaje de error.	

Tabla 19. HU Generar gráfico de polígono de frecuencia

Historia de Usuario	
Número: 8	**Nombre**: Crear gráfico de polígono de frecuencia
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada**: 2

Prioridad en el negocio: Alta	**Puntos estimados**: 1
Riesgo de desarrollo: Alto	**Puntos reales:** 1
Descripción: Permite la creación de un gráfico de polígono de frecuencia a partir de la información almacenada y al mismo se tiempo pueden modificar sus características de acuerdo a la opción que se seleccione en la interfaz de la aplicación.	
Observaciones: En caso de no existir variables de acuerdo a las características de un polígono de frecuencia se muestra un mensaje de error.	

Tabla 20. HU Generar gráfico de burbujas

Historia de Usuario	
Número: 9	**Nombre:** Crear gráfico de burbujas
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada**:2
Prioridad en el negocio: Alta	**Puntos estimados:** 1
Riesgo de desarrollo: Alto	**Puntos reales**: 1
Descripción: Permite la creación de un gráfico de burbujas a partir de la información almacenada y al mismo tiempo se pueden modificar sus características de acuerdo a la opción que se seleccione en la interfaz de la aplicación.	
Observaciones: En caso de no existir variables de acuerdo a las características de un gráfico de burbujas se muestra un mensaje de error.	

Tabla 21. HU Generar gráfico de Gantt

Historia de Usuario

Número: 10	**Nombre:** Crear gráfico de Gantt
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada:** 3
Prioridad en el negocio: Alta	**Puntos estimados:** 1
Riesgo de desarrollo: Alto	**Puntos reales:** 1
Descripción: Permite la creación de un gráfico de Gantt a partir de la información almacenada y al mismo tiempo se pueden modificar sus características, de acuerdo a la opción que se seleccione en la interfaz de la aplicación.	
Observaciones: En caso de no existir variables de acuerdo a las características de un gráfico de Gantt se muestra un mensaje de error.	

Tabla 22. HU Generar gráfico de Pareto

Historia de Usuario	
Número: 11	**Nombre:** Crear gráfico de Pareto
Cantidad de modificaciones: ninguna	
Usuario: Especialista de la CGR	**Iteración asignada**:3
Prioridad en el negocio: Alta	**Puntos estimados:** 1
Riesgo de desarrollo: Alto	**Puntos reales:** 1
Descripción: Permite la creación de un gráfico de Pareto a partir de la información almacenada y al mismo tiempo se pueden modificar sus características, de acuerdo a la opción que se seleccione en la interfaz de la aplicación.	
Observaciones: En caso de no existir variables de acuerdo a las características de un gráfico de Pareto se muestra un mensaje de error.	

Anexo 4: Tareas de ingeniería

Tabla 23. TI Generar gráfico de sectores

Tarea de Ingeniería	
Número de la tarea: 3	**Número de la HU:** 3
Nombre de la tarea: Crear gráfico de sectores	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 9/2/2015	**Fecha fin**: 15/2/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico de sectores se llama al método crearGraficoDeSectoresPorSerie (ArrayList<String> nombresColumnas, ArrayList<LinkedList<String>> columnasDeValores, LinkedList<String> columnaDeSerie, String titulo, String etiquetaEjeY, String etiquetaEjeX, boolean vertical, boolean en3D, boolean leyenda, boolean tooltips) de la clase **Graficadora.java**.	
Observaciones	

Tabla 24. TI Generar gráfico de línea

Tarea de Ingeniería	
Número de la tarea: 4	**Número de la HU:** 4
Nombre de la tarea: Crear gráfico de línea	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 16/2/2015	**Fecha fin**: 22/2/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico de línea se llama al método crearGraficoDeLineaPorSerie (ArrayList<String>	

nombresColumnas, ArrayList<LinkedList<String>> columnasDeValores, LinkedList<String> columnaDeSerie, String titulo, String etiquetaEjeY, String etiquetaEjeX, boolean vertical, boolean en3D, boolean leyenda, boolean tooltips) de la clase **Graficadora.java.**
Observaciones

Tabla 25. TI Generar gráfico de área

Tarea de Ingeniería	
Número de la tarea: 5	**Número de la HU:** 5
Nombre de la tarea: Crear gráfico de área	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 23/2/2015	**Fecha fin**: 1/3/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico de sectores se llama al método crearGraficoDeAreaPorSerie (ArrayList<String> nombresColumnas, ArrayList<LinkedList<String>> columnasDeValores, LinkedList<String> columnaDeSerie, String titulo, String etiquetaEjeY, String etiquetaEjeX, boolean vertical, boolean en3D, boolean leyenda, boolean tooltips) de la clase **Graficadora.java.**	
Observaciones	

Tabla 26. TI Generar gráfico de dispersión

Tarea de Ingeniería	
Número de la tarea: 6	**Número de la HU:** 6
Nombre de la tarea: Crear gráfico de dispersión	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1

Fecha de inicio: 11/03/2015	**Fecha de inicio**: 18/03/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico de dispersión se llama al método crearGraficoDeDispersion (ArrayList<String> nombresColumnas, ArrayList<LinkedList<String>> columnasDeValores, LinkedList<String> columnaDeSerie, String titulo, String etiquetaEjeY, String etiquetaEjeX, boolean vertical, boolean en3D, boolean leyenda, boolean tooltips) de la clase **Graficadora.java.**	
Observaciones	

Tabla 27. TI Generar gráfico de histograma

Tarea de Ingeniería	
Número de la tarea: 7	**Número de la HU: 7**
Nombre de la tarea: Crear gráfico de histograma	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 19/01/2015	**Fecha fin**: 26/03/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico de histograma se llama al método crearHistogramaSectoresPorSerie (ArrayList<String> nombresColumnas, ArrayList<LinkedList<String>> columnasDeValores, LinkedList<String> columnaDeSerie, String titulo, String etiquetaEjeY, String etiquetaEjeX, boolean vertical, boolean en3D, boolean leyenda, boolean tooltips) de la clase **Graficadora.java**	
Observaciones	

Tabla 28. TI Generar gráfico de polígono de frecuencia

Tarea de Ingeniería

Número de la tarea: 8	**Número de la HU:** 8
Nombre de la tarea: Crear gráfico de polígono de frecuencia	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 27/03/2015	**Fecha fin**: 3/04/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico polígono de frecuencia se llama al método crearPoligonoDeFrecuenciaPorSerie (ArrayList<String> nombresColumnas, ArrayList<LinkedList<String>> columnasDeValores, LinkedList<String> columnaDeSerie, String titulo, String etiquetaEjeY, String etiquetaEjeX, boolean vertical, boolean en3D, boolean leyenda, boolean tooltips) de la clase **Graficadora.java**	
Observaciones	

Tabla 29. TI Generar gráfico de burbujas

Tarea de Ingeniería	
Número de la tarea: 9	**Número de la HU:** 9
Nombre de la tarea: Crear gráfico de burbujas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 4/04/2015	**Fecha fin**: 11/04/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico polígono de frecuencia se llama al método crearGraficoDeBurbujaPorSerie (LinkedList<String> columnaEjeX, LinkedList<String> columnaEjeY, ArrayList<LinkedList<String>> listaColumnasDeValores, ArrayList<String> nombreColumnas, String titulo, String etiquetaEjeX, String etiquetaEjeY, boolean vertical, boolean leyenda, boolean tooltips) de la clase **Graficadora.java.**	
Observaciones	

Tabla 30. TI Generar gráfico de Gantt

Tarea de Ingeniería	
Número de la tarea: 10	**Número de la HU:** 10
Nombre de la tarea: Crear gráfico de Gantt	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 12/04/2015	**Fecha de inicio**: 19/04/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico de barras se llama al método crearDiagramaDeGantt (LinkedList<String> columnaDeTareas, String titulo, String etiquetaEjeX, String etiquetaEjeY, String etiqPlanificado, LinkedList<String>columnaDeInicio_FechasPlanific,LinkedList<String>columnaDeInicio_FechasReales, LinkedList<String> columnaDeFin_FechasPlanific, LinkedList<String> columnaDeFin_FechasReales, String etiqReal, boolean vertical, boolean leyenda, boolean tooltips) de la clase **Graficadora.java**	
Observaciones	

Tabla 31. TI Generar gráfico de Pareto

Tarea de Ingeniería	
Número de la tarea: 11	**Número de la HU:** 11
Nombre de la tarea: Crear gráfico de Pareto	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 19/04/2015	**Fecha de inicio**: 26/04/2015
Desarrollador responsable: Ana del Carmen Espinosa Robert	
Descripción: Para crear un gráfico de Pareto se llama al método crearGraficoDePareto(LinkedList<String>	

columnaDeCategoria, LinkedList<String> columnaDeValores, String etiquetaPorcAc, String etiquetaPorciento, String titulo, String etiquetaEjeX, String etiquetaEjeY, boolean vertical, boolean en3D, boolean leyenda, boolean tooltips） de la clase **Graficadora.java,**
Observaciones

Anexo 5: Carta de aceptación

UCI Universidad de las Ciencias Informáticas

CARTA DE ACEPTACIÓN

En cumplimiento con la fase de desarrollo y en función de la ejecución de la tesis: Sistema de Auditoría de Datos: Módulo de Gráficos v2.0, se hace entrega de los productos que se relacionan a continuación:

- CMódulo de Gráficos v2.0 (código fuente).

Entrega	Recibe
Nombre y apellidos: Ana del Carmen Espinosa Robert	**Nombre y apellidos:** Janier José Ramírez Landaburo
Cargo: Tesista	**Cargo:** Jefe de Proyecto AUDAT
Firma:	**Firma:**

UCI Universidad de las Ciencias Informáticas Decanato Facultad

Fecha: 15/06/2015

Fig. 33 Carta de aceptación

GLOSARIO DE TÉRMINOS

Variables cualitativas: se refieren a características o cualidades que no pueden ser medidas con números.

Variables cualitativas ordinales: presenta modalidades no numéricas que no admiten un criterio de orden**.** Ej: el estado civil con las siguientes modalidades: soltero, casado, separado, divorciado y viudo.

Variables cualitativas nominales: presenta modalidades no numéricas en las que existe un orden, por ejemplo la nota en un examen: 2, 3, 4 o 5, el puesto conseguido en una prueba deportiva: 1º, 2º, 3º o las medallas de una prueba deportiva: oro, plata, bronce.

Variables cuantitativas: se refieren a características o cualidades que pueden ser medidas con números.

Variables cuantitativas continuas: se refieren a los variables que pueden tomar cualquier valor (un número infinito de valores) dentro de un cierto intervalo.

Variables cuantitativas discretas: se refieren a las variables que sólo puede tomar valores dentro de un conjunto numerable, es decir, no acepta cualquier valor sino sólo aquellos que pertenecen al conjunto.

Interactivo: Se denomina a todo aquello que proviene o procede de interacción. La palabra interacción se conoce como aquella acción que se ejerce de manera recíproca entre dos o más sujetos, objetos, agentes, fuerzas o funciones.

CSV: Archivo de datos que contienen conjuntos separados por comas, donde cada nueva línea representa una fila nueva base de datos, y cada fila de base de datos tiene una o más campos separados por una coma, se pueden organizar en las células mediante un programa de hoja de cálculo o se inserta en una base de datos. Se utilizan comúnmente para la transferencia de datos entre bases de datos en un formato simple basado en texto