Universidad de las Ciencias Informáticas

Facultad 9

Graficador de datos en el plano para simulador de procesos.

TRABAJO PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

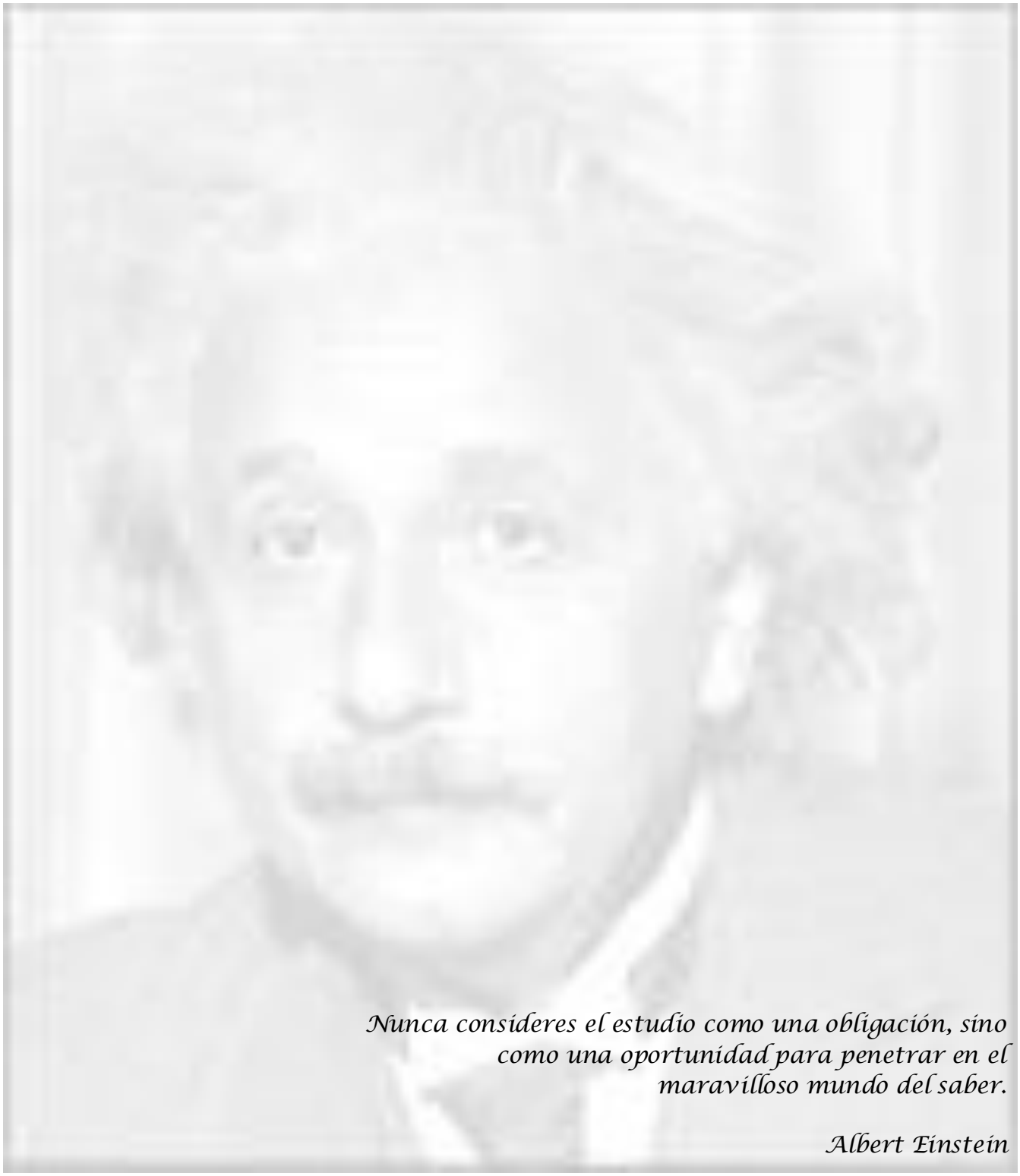
Autores:

Randy Aguiar Guerra

Luis Manuel Refeca González

Tutor: Ing Arnaldo Gandol

Ciudad de la Habana 4 de julio del 2008



Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el maravilloso mundo del saber.

Albert Einstein

DEDICATORIA

A toda mi familia, en especial a Mami, Papucho, Andy, Nina y Poder

A mi novia Lisandra.

Randy

A todas las personas que han aportado su granito de arena para que hoy yo pudiera graduarme… A toda mi familia que me dio el apoyo que necesitaba, en especial a mi madre, a mi hermano Carlos que es mi tesoro, a mis otros dos hermanos Miguel y Damián que los adoro, a mis abuelos y mis tíos a Susana que me educó con todo el amor del mundo y me dio siempre su apoyo de madre, de amiga y de hermana; a ella le debo casi todo lo que soy. Por último quiero dedicar este trabajo a la persona que se lo ha ganado junto conmigo, es la persona que nunca descansó en su afán de hacer de mí un hombre de bien y que vive tan orgulloso de mí, como yo de él… a Juan Manuel Refeca Cedeño, mi papá…

Luis Manuel

AGRADECIMIENTOS

A nuestros padres y familiares.

A todas las personas que en el transcurso de estos años nos han guiado y brindado sus conocimientos.

A nuestros amigos que siempre estuvieron ahí y a los que no pudieron estar.

A nuestro tutor Arnaldo Gandol Álvarez por su apoyo durante el desarrollo de este trabajo.

A la comunidad de software libre, especialmente de Java por sus múltiples aportes para el desarrollo de la aplicación.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de este trabajo y autorizamos a la Facultad 9 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Luis Manuel Refeca González

_____

Firma del autor

Randy Aguiar Guerra

_____

Firma del autor

Ing. Arnaldo Gandol Álvarez

_____

Firma del tutor

OPINIÓN DEL TUTOR

RESUMEN

En la actualidad existe un gran número de sistemas que se utilizan para la graficación de datos, muchos de los cuales no pueden ser utilizados con libertad pues se necesita de licencias y no se conoce la estructura que los respalda, por lo que su utilización se ve restringida solamente a las funcionalidades específicas para las que fueron creados.

En la Universidad de las Ciencias Informáticas existe el Polo Científico de Simulación en el cual la utilización de graficadores de datos es de gran apoyo para los softwares que en éste se desarrollan. Precisamente estos softwares utilizan un graficador como los descritos anteriormente, que el código no es libre y por lo tanto esto limita la expansión del producto que lo utiliza. Es por ello que el presente trabajo va encaminado a desarrollar un graficador de datos en el plano para su utilización en los distintos softwares de simulación que se desarrollen en el polo científico.

La presente investigación está basada en un estudio teórico sobre el tema para profundizar los conocimientos en la rama de la graficación y de la tecnología Java la cual fue utilizada para desarrollar la aplicación en cuestión, además de los detalles que guiaron la construcción del sistema.

Para el desarrollo de la aplicación se uso el Visual Paradigm como herramienta CASE profesional, la cual tolera el ciclo de vida completo del desarrollo del software y además contribuye a construir aplicaciones de calidad más rápido, mejores y a un menor coste. El lenguaje de programación que se utilizó fue el Java que proporciona un modelado de objetos más simple que C y C++ y elimina herramientas de bajo nivel que suelen conducir a errores, como son la manipulación directa de punteros. La plataforma utilizada fue la J2SE (Plataforma Java 2, Edición estándar) que es de distribución gratuita y proporciona una importante funcionalidad que es una Interfaz Gráfica de Usuario (GUI) por sus siglas en inglés. Como entorno de desarrollo integrado fue seleccionado el NetBeans 6.0 que es un IDE de código abierto que soporta el desarrollo de todos los tipos de aplicaciones en J2SE y que permite escribir, compilar, depurar y ejecutar programas en Java.

Después de analizar las necesidades del software que ya existe en el polo y estudiar los distintos tipos de gráficas que se pueden construir con los datos resultantes del proceso de simulación, se determinó que el sistema debería permitir la graficación en barras, líneas y pastel.

Resumen

El resultado final es una aplicación de escritorio con una interfaz de usuario donde se gestionan todos los datos necesarios para la graficación y que muestra los tres tipos de gráficas que se decidieron implementar.

Índice de Figuras.

Índice de Tablas

ÍNDICE

Introducción

El hombre siempre ha tratado de representarse las situaciones de su vida en un marco más reducido con el objetivo de hacer más sencilla su comprensión y minimizar riesgos. Con la aparición de la computación y las facilidades que esta brinda se comenzó a adentrar con más eficiencia y con mayores posibilidades de estudio y desarrollo en este mundo. En la actualidad son muchos los sistemas que existen y que diariamente se desarrollan con el fin de representar cualquier tipo de evento o situación de la vida diaria. El auge en este campo se debe precisamente a las grandes facilidades y las ventajas que conlleva la representación de los eventos reales en modelos y sistemas, por ejemplo: se puede usar para analizar y sintetizar una compleja y extensa situación real, permite el ahorro de materiales y de los elementos que conlleve el proceso, nos da una idea lo más cerca a la realidad del proceso simulado en cuestión y es en algunos casos es el único método disponible.

Como se ha expresado anteriormente son muchos los campos en los que podemos encontrar la simulación como herramienta de aprendizaje, estimación y desarrollo de procesos. Existe una gran diversidad en cuanto a las ramas donde podemos encontrar sistemas simuladores. Están presentes en la industria, en el perfil militar, en la electrónica, en la aviación, en la astronomía y en general en cualquier rama que lo permita.

Los simuladores pueden tener como una herramienta de utilidad un graficador, el cual se utiliza dentro del software para observar con mayor facilidad los distintos comportamientos que puedan tener ciertas variables que intervengan en el proceso de simulación, así como para la más fácil comparación entre ellas en el transcurso de los procesos.

La representación de datos de forma gráfica es muy útil para una mejor comprensión de éstos, para el análisis final del funcionamiento de los mismos y para el mejor entendimiento de las conclusiones a las que se pretenda arribar. También hay que saber seleccionar adecuadamente lo que se va a representar y de que forma se va a hacer para que los datos puedan ser interpretados de forma adecuada.

La representación de estos datos se realizan en una amplia gama de gráficas, las cuales tienen sus propias características y diseño, cada una se utiliza en dependencia de la situación y de los datos a representar, garantizando así el mejor entendimiento de los mismos en cada situación.

La gran mayoría de los simuladores que se utilizan en al actualidad tienen, como una herramienta muy útil para el mejor entendimiento de los procesos que ocurren, a un graficador para la representación de los datos resultantes del proceso de simulación de una forma más cómoda para su análisis y comprensión. Actualmente existe gran cantidad de graficadores que están desarrollados en diversos lenguajes y que son utilizados en muchas aplicaciones como es el caso de Microsoft Excel. Muchos de estos graficadores necesitan de una licencia para su uso y carecen de algunas funcionalidades que son útiles para el uso de ellos en software de simulación.

La facultad #9 de la Universidad de las Ciencias Informáticas esta inmersa en el desarrollo de un Polo Científico de Simulación, y no cuenta con un graficador propio para su utilización en los diferentes productos a desarrollar, por lo que resulta una limitación a la hora de su uso o de redefinir algunas funcionalidades que quieran ser cambiadas. Teniendo en cuenta esta situación que se presenta se ha planteado como **problema científico** la inexistencia de un graficador propio de datos en el plano sobre software libre para la representación de los resultados en la simulación de procesos en la Universidad de las Ciencias Informáticas, lo cual conlleva a tener como **objetivo** desarrollar un graficador de datos en el plano para su utilización en simuladores de procesos.

La investigación estará enfocada hacia el Proceso de Graficación de Datos como **objeto de estudio** y específicamente dentro de él al **campo de acción** Graficación de Datos en dos Dimensiones.

La **idea a defender** es la siguiente: con el desarrollo de un graficador de datos en el plano para simuladores de proceso, se dará una solución al problema existente para la graficación de datos, permitiendo utilizar este producto con plena libertad en los softwares de simulación en la Universidad de las Ciencias Informáticas

Para el satisfactorio cumplimiento del objetivo se plantean las siguientes tareas:

1. Estudiar los diferentes tipos de gráficas de dos dimensiones.
2. Seleccionar los tipos de gráficas más adecuados para el graficador a desarrollar.
3. Seleccionar el lenguaje de modelado y la herramienta informática más adecuada para crear los artefactos del sistema durante su ciclo de vida.
4. Estudiar las tecnologías para la implementación del producto y seleccionar la más adecuada.

5. Análisis y diseño de una arquitectura adecuada para el desarrollo del graficador.
6. Implementación del graficador de datos en el plano.

Es de importancia destacar que hasta el momento no hay ninguna referencia de la existencia de un graficador que haya sido desarrollado para su utilización específica en algún software de simulación, pero como se ha planteado existe un gran número de ellos, desarrollados para su utilización en diversas ramas como son por ejemplo: .netCHARTING, PDAGraphiX2, SwiftChart y las librerías ZedGraphics que son las que actualmente se están utilizando para la graficación de datos en el simulador de procesos que existe en el Polo de Simulación de la Universidad de las Ciencias Informáticas.

Los métodos utilizados para el desarrollo de la investigación son los siguientes:

- **Histórico-lógico**: Se hace un estudio de los diferentes graficadores que existen, al igual que de las distintas gráficas que ellos utilizan para el posterior desarrollo eficiente del graficador.
- **Analítico – sintético:** Se analizan los diferentes tipos de gráficas que existen y sus características lo que permite determinar elementos comunes entre ellas y establecer una relación para un mejor desarrollo de la solución propuesta.
- **Inductivo – deductivo:** Se estudian las características de algunos graficadores, sus principales funcionalidades y cómo son utilizados, para determinar que es lo que los identifica y determinar comportamientos comunes, así como para a partir de elementos generales llegar a conclusiones particulares de cada uno de ellos.
- **La entrevista:** realizadas a líderes de proyecto para obtener información sobre las herramientas de graficación que utilizan los proyectos productivos en la universidad.

Capítulo 1. Fundamentación Teórica.

1.1 Introducción.

En este capítulo se hace una referencia a los conceptos fundamentales asociados con el dominio del problema, se aborda de forma detallada el objeto de estudio de la investigación, se realiza una descripción detallada y argumentada de la situación problemática y se enumeran algunos ejemplos de sistemas que son utilizados para la graficación de datos.

1.2 Conceptos asociados al dominio del problema.

1.2.1 Simulación.

La simulación no es más que la representación de un fenómeno a través de modelos, lo que permite analizar sus características con mayor facilidad sin tener que desarrollar el fenómeno, con lo que se ahorra tiempo y recursos, uno de los objetivos primordiales de una simulación es analizar los resultados para así conocer con anterioridad su comportamiento y en caso posible mejorarlos en el momento que se lleve a cabo el fenómeno en la vida real.(1)

Thomas H. Naylor y R. Bustamante la definen así: "Simulación es una técnica numérica para conducir experimentos en una computadora digital. Estos experimentos comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real a través de largos períodos de tiempo".(2)

Una definición más formal formulada por **R.E. Shannon** es: "La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias -dentro de los limites impuestos por un cierto criterio o un conjunto de ellos - para el funcionamiento del sistema".(2)

Es importante destacar que a pesar de las ventajas que nos ofrece la simulación, existen algunas desventajas debido a que con frecuencia el proceso de desarrollar un modelo es largo y complicado por lo que un buen modelo de simulación puede resultar muy costoso o se necesita emplear un período de tiempo grande para su desarrollo, por lo que se recomienda estudiar bien la situación antes de lanzarse a modelarlo.

1.2.1.1 Sistemas de simulación.

Para la representación de eventos en sistemas de simulación se tienen en cuenta las cualidades y particularidades de cada uno de ellos y según esto los diferentes sistemas en que se agrupan pueden ser los siguientes(3):

Sistema discreto: Las variables de estado cambian solo en puntos discretos o contables en el tiempo. Un ejemplo típico de simulación discreta ocurre en las colas donde estamos interesados en la estimación de medidas como la longitud de la línea de espera. Tales medidas solo cambian cuando un cliente entra o sale del sistema; en todos los demás momentos, no ocurre nada en el sistema desde el punto de vista de la inferencia estadística.

Sistema continuo: Las variables de estado cambian en forma continua a través del tiempo. Un ejemplo típico de simulación continua es el estudio de la dinámica de la población mundial.

Modelo estático de simulación: Representación de un sistema en determinado punto en el tiempo.

Simulación dinámica: Representación de como evoluciona un sistema a través del tiempo.

1.2.2 Graficador.

Un graficador es un registrador que presenta en forma de gráfica la información de salida, indicando las tendencias. Los dispositivos gráficos se utilizan comúnmente en la actualidad, junto con las computadoras, para tener un registro permanente del material que se está procesando en una computadora.(4)

1.2.3 Proceso de graficación.

En la actualidad es muy infrecuente encontrar un gráfico hecho a mano. Generalmente se emplean sistemas graficadores de microcomputadoras. Esto no invalida la necesidad de conocer las reglas y convenciones establecidas con respecto a la confección de los mismos. Dada la enorme libertad que brindan algunos de esos sistemas, en más de una oportunidad hemos visto gráficos confeccionados por estos medios que presentan errores, entre otras cosas, por seleccionar un tipo de gráfico no adecuado para la información que se desea representar.

Entonces se puede decir que el proceso de graficación no es más que los pasos necesarios que se ejecutan para la confección de una gráfica a través de un graficador.

1.3 Objeto de Estudio.

El proceso de graficación de datos es un poco complejo debido a la serie de condiciones que se deben tener en cuenta para llevar dichos datos a una representación gráfica que sea aceptada por los usuarios finales. Según estudios realizados conviene utilizar frases para relaciones entre 2 o 3 datos, tablas entre 3 y alrededor de 20 datos y gráficos a partir de 3 relaciones y tanto más, cuanto mayor sea el número de datos y relaciones entre ellos.(5)

El mensaje anterior se puede observar en la siguiente tabla.

	OK
Frases	Relaciones entre 2 ó 3 datos.
Tablas	Relaciones entre 3 y 20 datos (aproximadamente)
Gráficos	Desde 2 en adelante. Especialmente útiles para grandes cantidades de datos.

Tabla 1 Relaciones entre datos

Pero podríamos utilizar también un gráfico para expresar lo mismo (siguiente figura).

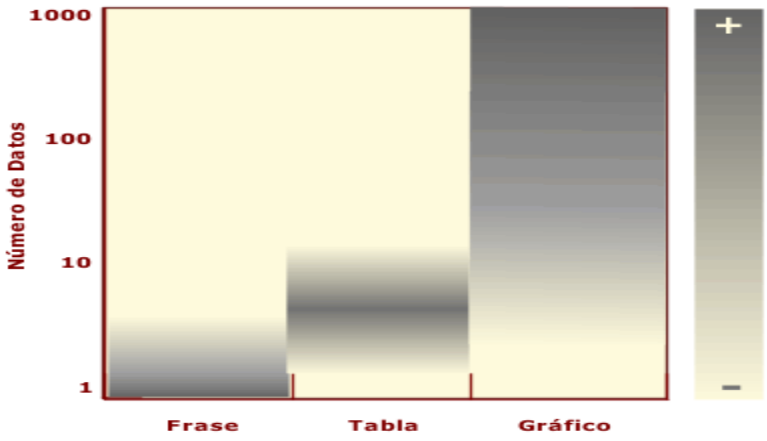


Figura 1 Gráfica de relaciones

1.3.1 Descripción General.

No es fácil destilar unos principios elementales y generales que resuman la buena práctica de realizar gráficos para la representación de los datos. Los objetivos fundamentales por los que se realiza la graficación son los siguientes(6):

- **Comunicar un mensaje:** La producción ha aumentado pero aún estamos por debajo de lo planificado.
- **Presentar grandes cantidades de información:** Es más fácil de entender un gráfico que una tabla llena de números representados en filas y columnas.
- **Revelar los datos:** Descubrir las relaciones de causa-efecto, sabes que es lo que pasa.
- **Controlar la evolución:** Conocer como se van comportando los parámetros de forma periódica.

En definitiva la representación gráfica de datos es una forma más fácil de comprender una información que se desea brindar a las personas.

1.3.1.1 Graficación en 3D.

Un gráfico en tres dimensiones se origina mediante un proceso de cálculos matemáticos sobre entidades geométricas tridimensionales y su propósito es conseguir una proyección visual en dos dimensiones para ser mostrada en una pantalla o impresa en papel. La graficación en 3D hace que los gráficos resultantes tengan una apariencia que da la idea de un mejor acabado en el gráfico, pero este resulta más engorroso a la hora de leerlo para apreciar sus datos y que se dificulte extraer las diferencias entre los valores en cuestión. Los gráficos en 3D tienen usos muy limitados (solo en el ejemplo de superficies están recomendados). Mejor utilizar gráficos en 2D y así el mensaje quedará más claro y fácil de entender.

1.3.1.2 Graficación en 2D.

La graficación en 2D es utilizada para la representación de los datos en dos dimensiones. Esta forma de representar la información de forma gráfica es utilizada en la vida diaria por un amplio número de instituciones para la comprensión más rápida de los datos emitidos y así a su vez que se transmitan esta comprensión de datos de forma más rápida a otras personas involucradas en las actividades en cuestión. Algunas empresas usan estas técnicas de visualización para aumentar o resumir sus informes tradicionales. Algunos de los gráficos más simples para representar datos en dos

dimensiones son los de línea, columnas, puntos, barras, pastel, los cuales han sido utilizados desde hace tiempo.

1.3.1.3 Algunos tipos de gráficas.

1.3.1.3.1 Gráfica de barra.

Una gráfica de barras es aquella representación gráfica bidimensional en que los objetos gráficos elementales son un conjunto de rectángulos dispuestos paralelamente de manera que la extensión de los mismos es proporcional a la magnitud que se quiere representar.(7)

Se usan cuando se pretende resaltar la representación de porcentajes de datos que componen un total. Una gráfica de barras contiene barras verticales que representan valores numéricos, generalmente usando una hoja de cálculo. Las gráficas de barras son una manera de representar frecuencias. Las frecuencias están asociadas con categorías. Una gráfica de barras se presenta de dos maneras: horizontal o vertical. El objetivo es poner una barra de largo (alto si es horizontal) igual a la frecuencia. La gráfica de barras sirve para comparar y tener una representación gráfica de la diferencia de frecuencias o de intensidad de la característica numérica de interés.

A continuación se muestran algunos ejemplos para la mejor comprensión de todo lo abordado anteriormente:

Gráfica de barra donde los elementos del eje horizontal están representados porcentualmente.

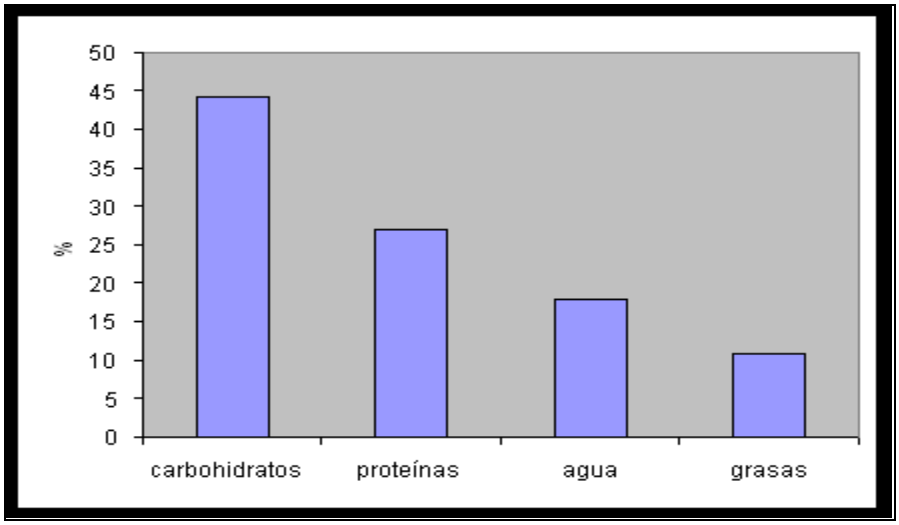


Figura 2 Gráfica de barra porcentual

Gráfica de barra donde se representan varios elementos en el eje horizontal y sus valores correspondientes en el eje vertical.

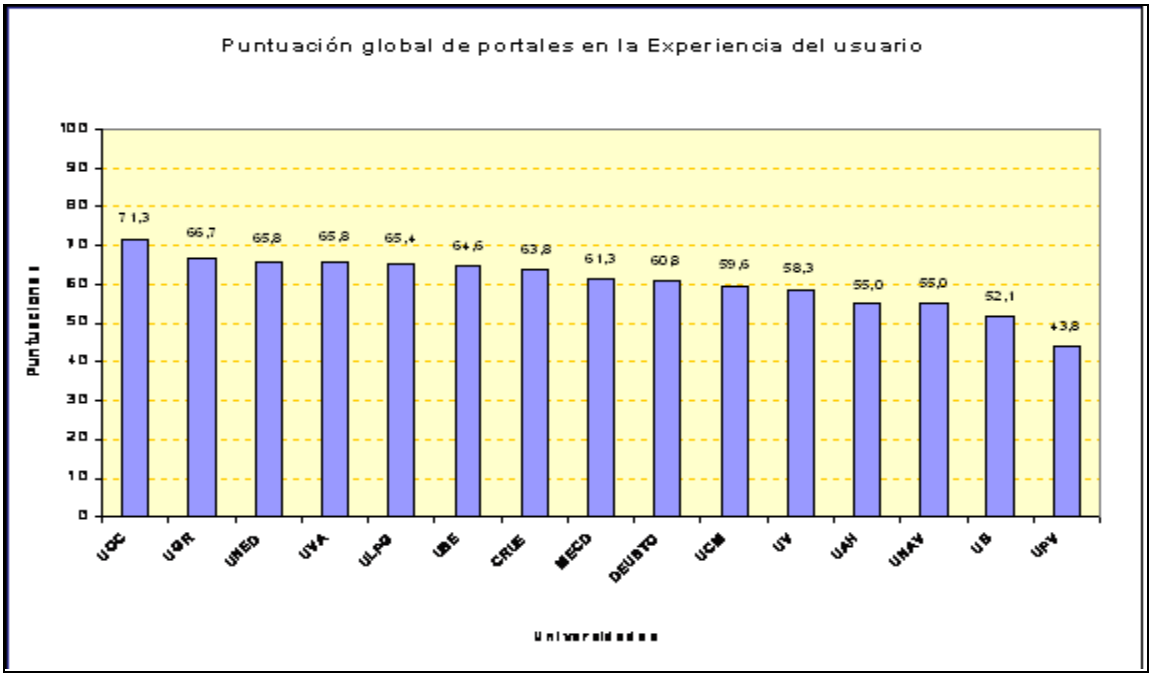


Figura 3 Gráfica de barra con valores reales

También existen otras gráficas de barras llamadas **gráficas de barras agrupadas**, los cuales contiene varias series de datos**,** por ejemplo las ventas mensuales en varios países. En este caso el eje secuencial contendría los meses y el cuantitativo la cifra de ventas. Cada serie de datos se representa mediante un conjunto de rectángulos que comparten color o textura.

En el siguiente ejemplo la gráfica de barras agrupadas representa la cantidad de personas abastecidas por suministro público y por abastecimiento propio en el período 1955-2000 en los Estados Unidos de América.

Figura 4 Gráfica de barras agrupadas.

Como se ha podido observar en los ejemplos anteriores una gráfica de barras cuenta con los siguientes elementos:(7)

Un eje cuantitativo con una escala lineal que sirve de referencia a la magnitud de la variable en cuestión. En un gráfico de columnas éste es el eje de ordenadas [**Y**] y en uno de barras es el eje de abscisas [**X**]). Este eje puede contener valores negativos.

Un eje categórico u ordinal en el que se disponen las categorías o los elementos de la secuencia (el eje de abscisas [**X**] en un gráfico de columnas o el de ordenadas [**Y**] en un gráfico de barras). Este eje

es perpendicular al cuantitativo. En el caso de las gráficas de barras agrupadas en el eje horizontal estarán representadas las categorías cualitativamente mediante **series**, también conocidas como períodos.

Un conjunto de rectángulos cuya extensión paralela al eje cuantitativo es proporcional a la magnitud de la categoría o secuencia representada en el eje.

Es importante destacar que las gráficas de barras tienen una amplia aplicación en diferentes sectores, pues se pueden utilizar de diversas formas y en muchas variantes. Los ejemplos expuestos anteriormente muestran las más usadas, pero existe otro gran número de variantes de representar datos en forma de barra, principalmente combinaciones de los ejemplos anteriores.

1.3.1.3.2 Gráfica de líneas.

Las gráficas de líneas muestran una serie como un conjunto de puntos conectados mediante una línea. Los valores se representan por el alto de los puntos con relación al eje Y. Las etiquetas de las categorías se presentan en el eje X. Estas gráficas suelen utilizarse para comparar valores a lo largo del tiempo. (8)

Existen dos tipos de gráficas de líneas:

1-Gráfica de líneas.

Una gráfica de líneas muestra los parámetros como líneas rectas individuales, con un punto de datos por cada serie. El valor de los datos determina el alto de cada punto en el eje Y. Estas gráficas están compuestas por los siguientes elementos:

1-Valores: Los valores de un parámetro determinan el alto de las líneas de ese parámetro en el eje Y. Las etiquetas de los valores aparecen en el eje Y. Cada parámetro consta de un conjunto de valores que aparecen como una línea separada.

2-Grupos de series: Las series se muestran como etiquetas en el eje X. Varios grupos se anidan. En los gráficos de líneas, las series suelen estar relacionadas con el tiempo.

3-Grupos de parámetros: Las parámetros se muestran como líneas separadas en la gráfica. Cada parámetro también se presenta en la leyenda de la gráfica.

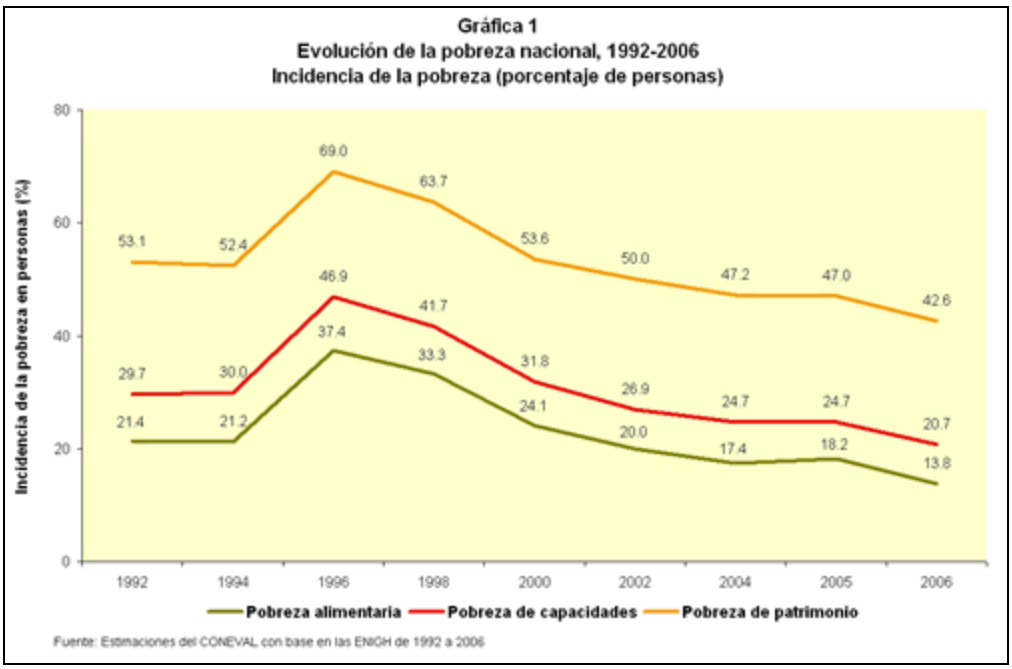


Figura 5 Gráfica de líneas de tres parámetros.

2- Gráfica de líneas suavizadas.

Una gráfica de líneas suavizadas presenta los parámetros como líneas separadas. Las líneas son curvas, en lugar de rectas. El valor determina el alto de cada columna. Esta gráfica esta compuesta por los siguientes elementos:(8)

1- Valores: Los valores de un parámetro determinan el alto de las líneas de ese parámetro. Las etiquetas de los valores aparecen en el eje Y. Los valores de cada parámetro aparecen como una línea separada.

2- Grupos de series: Las series se muestran como etiquetas en el eje X. Varios grupos se anidan. En los gráficos de líneas, las series suelen estar relacionadas con el tiempo.

3- Grupos de parámetros: Los parámetros se muestran como líneas separadas en la gráfica. Cada parámetro también se presenta en la leyenda de la gráfica.

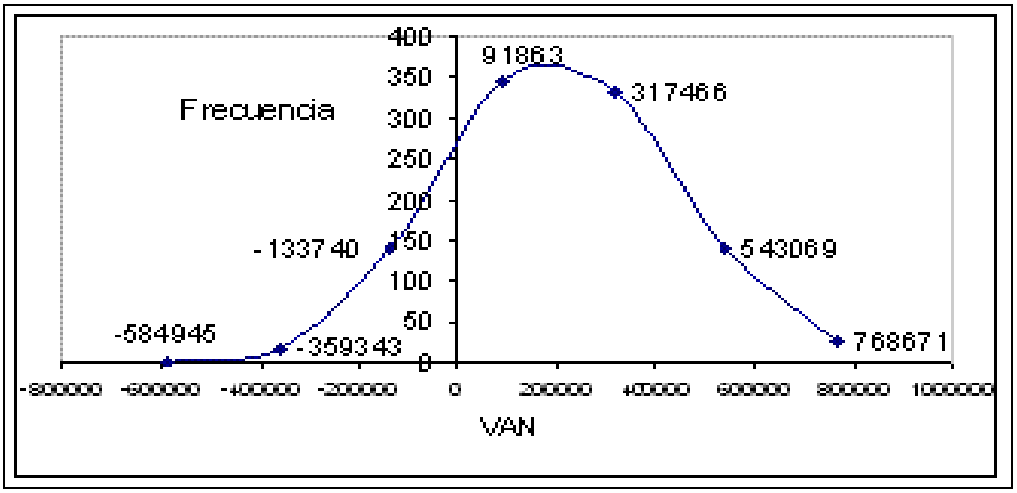


Figura 6 Gráfica de líneas suavizadas.

1.3.1.3.3 Gráfica de pastel.

Se usa, fundamentalmente , para representar distribuciones de frecuencias relativas(%) de una variable cualitativa o cuantitativa discreta .En esta gráfica se hace corresponder la medida del ángulo de cada sector con la frecuencia correspondiente a la clase en cuestión. Si los 360º del círculo representan el 100 % de los datos clasificados, entonces a cada 1% le corresponden 3,6º. Luego, para obtener el tamaño del ángulo para un sector dado bastaría con multiplicar el por ciento correspondiente por 3,6º (por simple regla de tres). (9)

Mediante un sector circular se representan las medidas angulares correspondientes a las diferentes categorías, respetando el orden establecido en la tabla, partiendo de un punto dado de la circunferencia. Ese punto dado generalmente es el punto más alto de la circunferencia. Si lo que se representa en cada sector no puede colocarse dentro del mismo, se elabora una leyenda o se coloca fuera, adyacente al mismo. Se acostumbra a diferenciar los sectores con tramas o colores diferentes, lo que hace que resulte un gráfico más vistoso que el de barras simples.

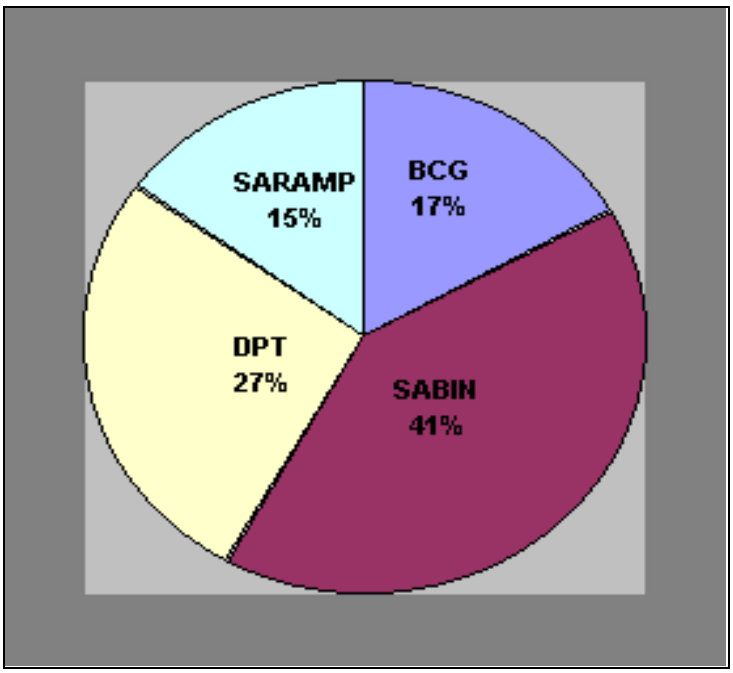


Figura 7 Gráfica de pastel.

1.3.1.3.4 Gráfica de burbuja.

La gráfica de burbujas muestra cada serie de valores como puntos de datos en el espacio del gráfico en función de los valores X e Y de la serie de valores. Un tercer valor, Size[1], determina el tamaño del símbolo del punto de datos. Aunque los grupos de categorías y series son opcionales, se requiere al menos uno de estos grupos para que el gráfico muestre datos significativos. (10)

Estas gráficas presentan los siguientes elementos:

1-Valores: Los valores de una serie de valores determinan la posición de cada punto de los datos del gráfico. Hay tres valores para cada serie: **X**, **Y** y **Size**. Estos tres valores son obligatorios. El valor **X** determina la posición del punto de datos en el eje X. El valor **Y** determina la posición del punto de

[1] Size (Inglés): hace referencia a la talla o tamaño.

datos en el eje Y. El valor **Size** determina el tamaño del símbolo del punto de datos. Si la gráfica contiene varias series de valores, tanto la gráfica como la leyenda mostrarán varias series.

2-Grupos de categorías: Los valores de categorías se presentan como puntos de datos dentro de un grupo de series.

3-Grupos de series: Las series se presentan como grupos de puntos de datos en el gráfico. Cada serie aparece de forma independiente en la gráfica (cada una con un color o símbolo diferente) y en la leyenda.

Este ejemplo compara la distancia que existe en cada uno de los planetas interiores de nuestro sistema solar al Sol contra el tiempo que necesitan para recorrer sus órbitas, y el tamaño de las burbujas indica la masa de cada planeta.

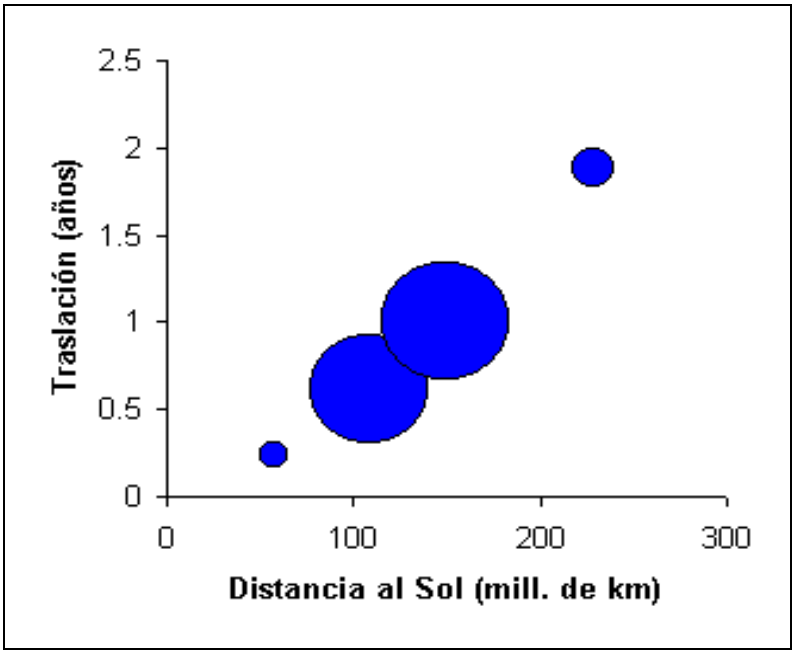


Figura 8 Gráfica de burbuja.

1.3.1.3.5 Gráfica de superficie.

Se pueden trazar datos que se organizan en columnas o filas de una hoja de cálculo en una gráfica de superficie. Una gráfica de superficie es útil cuando busca combinaciones óptimas entre dos conjuntos

de datos. Como en un mapa topográfico, los colores y las tramas indican áreas que están en el mismo rango de valores.

Puede utilizar una gráfica de superficie cuando ambas categorías y series de datos sean valores numéricos. A continuación un ejemplo gráfico para ilustrar de mejor manera lo expuesto.

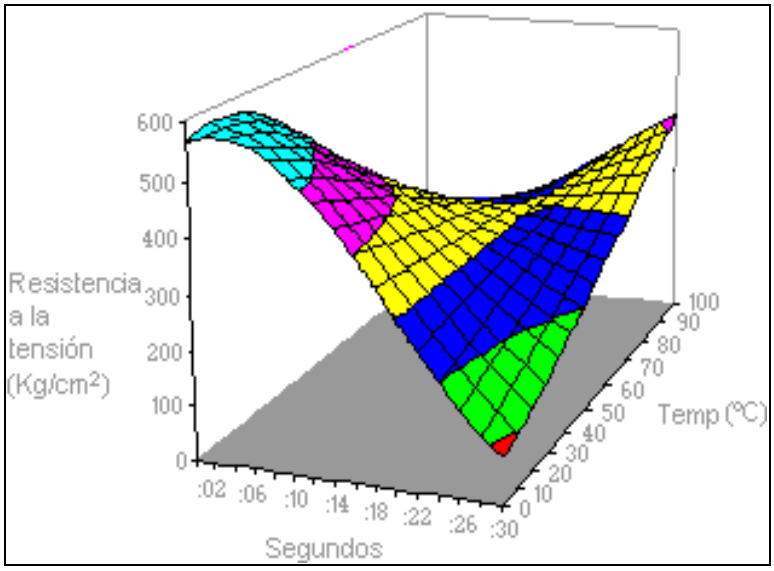


Figura 9 Gráfica de superficie.

Según lo expresado en (11) estos gráficos se dividen en los siguientes tipos:

Superficie 3D. Las gráficas de superficie 3D muestran tendencias en valores entre dos dimensiones de una curva continua. Los colores de una gráfica de superficie no representan las series de datos; representan la distinción entre los valores.

Trama de superficie 3D. Mostrado sin color, un gráfico de superficie 3D se denomina gráfico de trama de superficie 3D.

Contorno y trama de contorno. Las gráficas de contorno y trama de contorno son gráficas de superficie vistos desde arriba. En una gráfica de contorno, los colores representan rangos específicos de valores. Una gráfica de trama de contorno se muestra sin color.

1.3.2 Descripción actual del dominio del problema.

Para la realización del sistema que se pretende desarrollar es necesario indagar y conocer las particularidades del entorno en el cual se utilizará el graficador. Los simuladores de procesos son sistemas los cuales muestran la aproximación a la realidad de ciertos procesos que se llevan a cabo en la vida diaria y que sirven como instrumentos de aproximación y observación de los futuros comportamientos que puedan tener los procesos, así como para la mejora de los resultados que se observarán en estos últimos.

El actual simulador de procesos químicos de la industria azucarera cubana, el Sistema Termo Azúcar (STA 4.0.3) que se desarrolla en el Polo Científico de Simulación de la Universidad de las Ciencias Informáticas utiliza gráficas de barras, líneas y pastel debido a que estas son las más convenientes para la representación de los datos que se generan en el proceso de simulación.

1.3.3 Situación Problemática.

Actualmente existe una gran cantidad de graficadores desarrollados en distintos lenguajes, muchos de estos graficadores necesitan de una licencia para su uso y no se conoce su arquitectura por lo que el trabajo con ellos se hace dificultoso .Al no tener conocimientos sobre la arquitectura que fue utilizada para el desarrollo de estos graficadores entonces no se tiene la posibilidad de reestructurar algunas funcionalidades que estos posean y agregar otras en dependencia de las necesidades que se presenten en los distintos sistemas a desarrollar por el Polo científico de Simulación que está siendo desarrollado por la facultad 9 de la Universidad de las Ciencias Informáticas.

La situación planteada anteriormente provoca que sea dificultoso el trabajo con estos graficadores y en muchas ocasiones no cumplan a plenitud con ciertas funcionalidades que se pretendan utilizar en los distintos software .Con el desarrollo de un graficador propio de datos en el plano donde se tenga conocimiento de la arquitectura utilizada será más factible su utilización en los diferentes simuladores de procesos que se desarrollen y será posible agregar o modificar funcionalidades según en el software que se desee utilizar.

1.4 Análisis de otras soluciones existentes.

Un gran número de simuladores existentes que utilizan un graficador como una herramienta auxiliar que le permita a los usuarios de los mismos la mayor compresión de los datos resultantes de la simulación de los procesos que ellos representan, no utilizan un software de graficación que se haya

desarrollado específicamente para su uso en él; sino que utilizan otra herramienta de graficación que ya este desarrollada y que cumpla con ciertas funcionalidades que se deseen brindar a los usuarios que utilizarán el software.

Después de los estudios realizados en la Universidad de las Ciencias Informáticas no se tiene ninguna referencia de la existencia de un graficador que se haya desarrollado en algún polo científico de investigación para su uso en los productos que se desarrollan en esas áreas de trabajo, sino que utilizan la herramienta de esta naturaleza más afín con sus necesidades. Para un mejor estudio sobre este tema se realizaron algunas entrevistas a líderes de proyectos en la universidad.

A continuación se hace referencia a algunas de estas herramientas de graficación que ya existen.

1.4.1 .netCHARTING

El graficador .netCHARTING incluye una amplia gama de tipos de gráficas así como muchos tipos de combinaciones poderosas de gráficas de pastel, barras, columnas, líneas, puntos, radares y más, tanto en 2 dimensiones como en 3D.

.netCHARTING permite mostrar grandes cantidades de datos generados dinámica y fácilmente a través de una interfaz visual. El .netCHARTING esta construido con 100% de código administrado con el lenguaje C # y siempre con una amplia lista de ejemplos tanto en VB.NET[2] como C #, este alto rendimiento de gráficos de control también contiene una sección rica de acceso a datos y sistema de totalización con el apoyo de cálculo. Esta combinación única de características provee a los usuarios el acceso inmediato a sus datos existentes en un servidor.(12)

Esta solución que consta de muchas ventajas en lo referente al proceso de graficación de datos, posee **limitantes** para su utilización:

1. Está desarrollada sobre software propietario(C#.NET).
2. Necesita de una licencia para instalarlo y utilizarlo.

[2] VB.NET: se refiere al lenguaje de programación Visual Basic de la plataforma .NET.

1.4.2 PDAGraphiX2.

PDAGraphics es una excelente solución para evitar frustraciones a la hora de intentar generar gráficas mediante Pocket Excel. Una potente herramienta gráfica, flexible y dinámica que permite generar gráficas con facilidad.(13)

Gracias a su entorno desplegable y funcional en segundo plano, se puede, con la simple selección de las casillas y valores determinados, generar gráficas evolutivas al instante. Basado en la función gráfica de Excel para Windows, PDAGraphics soporta múltiples formatos de presentación.

Estas son sus principales características:

• Rápido sistema de generación de gráficas
• Modo landscape
• Sistema *Drag & Drop*
• Diferentes niveles de zoom disponibles
• Múltiples fondos disponibles
• Cambios de valores a tiempo real
• 100% compatible con Microsoft Pocket Excel
• Gráficas 3D
• Modo pantalla completa
• Iconos de acceso directo

En definitiva, PDAGraphics es una práctica y elaborada solución para corregir la carencia de generación de gráficas presente en Pocket Excel.

Aunque tiene muchas ventajas y funcionalidades para el desarrollo de la graficación de datos tanto en 2 dimensiones como en 3D, este software tiene grandes **limitantes** que impiden su utilización eficiente:

1. Está desarrollado sobre software propietario.
2. Se desconoce su arquitectura y no permite acceder a su código fuente para modificaciones que se deseen implementar.

3. Sus mejoras no son gratuitas.

1.4.3 SwiftChart.

Este graficador está desarrollado en Java y genera gráficas dinámicas de barras, línea, pastel, apiladas y agrupadas en 2D y 3D con un amplio rango de parámetros y tipos de gráficas. Principales características y tipos de gráficas desarrolladas en esta herramienta: Gráficas en 2D/3D, compatible con diferentes buscadores, control de parámetros de Java Script, etiquetas de información, detalles de información contextual, cálculo de tendencias, control de profundidad de 3D, Múltiples ejes Y, escala automática de gráficas. Gráfica principal y tipos de gráficas. Gráficas de barra en 2D/3D, gráficas de línea en 2D/3D, gráficas de pastel en 2D/3D, gráficas de barras apiladas en 2D/3D, gráficas de barras de porcentaje apiladas en 2D/3D, gráficas de barras horizontales en 2D/3D, gráficas área en 2D/3D, gráficas de porcentaje de área en 2D/3D, gráficas de XY esparcidas.

Al igual que las demás soluciones este software ofrece grandes ventajas pero también tiene algunas **limitantes**:

1. Su versión gratuita es solo por 30 días.
2. Necesita de una licencia que es pagada(14).

1.4.4 Librería ZedGraphics.

Esta librería posee muchas funcionalidades implementadas que permiten la graficación de datos en el plano de muchas formas. Actualmente el simulador STA (Sistema Termo Azúcar) del Polo Científico utiliza ZedGraphics para graficar los datos resultantes en el proceso de simulación. Dicha librería es de fácil uso y entre sus tipos de gráfica encontramos las de barras y las de líneas que son las más utilizadas.

Las principales **limitaciones** que tiene son:

1. Está desarrollada en software propietario.
2. No se conoce su arquitectura.

1.5 Conclusiones.

Al no contar con un graficador que pueda ser usado con libertad y modificado para adaptarlo a diferentes condiciones se decidió realizar un sistema en el cual estas situaciones no sean un inconveniente a la hora de su utilización, por lo que desde el punto de vista de los autores es de gran importancia todos los temas abordados en este capítulo pues brindan un conocimiento acerca de todos los conceptos asociados al dominio del problema y de la situación actual del entorno donde coexiste el problema a resolver. Se desarrolló un amplio estudio de los diferentes tipos de gráficas que existen y cuales son las más usadas, según sus características, para su desarrollo.

Capítulo 2.Tendencias y tecnologías actuales a desarrollar.

2.1 Introducción.

En el siguiente capítulo se argumentará sobre las tecnologías y herramientas que se usaron para el desarrollo del software. Se define el lenguaje de modelado a utilizar y sus ventajas, así como la metodología de desarrollo de software y el lenguaje de programación a utilizar con sus aspectos esenciales.

2.2 El software libre.

Software Libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más concreto, se refiere a cuatro libertades de los usuarios del software:(15)

1. La libertad de usar el programa, con cualquier propósito.
2. La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades. El acceso al código fuente es una condición previa para esto.
3. La libertad de distribuir copias, con lo que puedes ayudar a múltiples usuarios.
4. La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Software libre no significa no comercial. Un programa libre debe estar disponible para uso comercial, desarrollo comercial y distribución comercial. El desarrollo comercial del software libre ha dejado de ser inusual; el software comercial libre es muy importante.

En la actualidad hay una gran cantidad de empresas e instituciones que se relacionan con el mundo del software libre, ya sea porque utilizan software con estas características o que se dedican al desarrollo de programas libres. Muchos países han decidido entrar en este mundo y así tener libertades para distribuir, modificar y obtener el código fuente sin ninguna restricción.

Desde el año 2006 se produjo una noticia muy importante y esperada para el mundo de la programación, la empresa Sun dio a conocer que Java sería software libre. Esto es una buena noticia para todos los que programamos en Java, pues esto permite de alguna manera entender su

funcionamiento interno y llegar a mejorarlo. Además la licencia GPL[3] permite a todos los usuarios y empresas interesadas en sacar una propia versión de Java, la cuál puede ser mejorada u orientada a un aspecto específico de las actividades.

La licencia GPL es la licencia de software más libre que existe y con esto Sun pone fin a su hegemonía del lenguaje. Si Sun se le ocurriera dejar de dar soporte, podrían existir cientos de empresas que tomarían el código y lo podrían seguir desarrollando, garantizando con esto la seguridad para los desarrolladores de aplicaciones en este lenguaje.

2.3 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta.

En (16) UML es definido como un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

En 1997 sale a la luz el Lenguaje Unificado de Modelado en su versión 1.0 producto del trabajo conjunto de Grady Booch, James Rumbaugh, e Ivar Jacobson, los cuales intercambiaron conocimientos y lograron grandes avances en el campo de la modelación de sistemas orientados a objetos.

Se escogió el Lenguaje Unificado de Modelado como el lenguaje para la modelación de la solución propuesta porque este es un lenguaje expresivo, claro y fácil de utilizar y aprender, permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos, puede conectarse con lenguajes de programación (Ingeniería directa e inversa) y muestran gran legibilidad en el momento de interpretar los elementos que ellos muestran.

[3] GPL: se refiere a Licencia Pública General por sus siglas en inglés.

2.4 El Proceso Unificado de Desarrollo de Software (RUP) como base en el desarrollo de la solución.

Existen otras metodologías de desarrollo de software de gran éxito y utilización en la actualidad como son Extreme Programing (XP) y Microsoft Solution Framework (MSF). Una de las particularidades de XP es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto, además el cliente puede añadir, cambiar o quitar requerimientos en cualquier momento. Por su parte MSF es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Aunque el desarrollo del proyecto es relativamente pequeño se ha determinado la utilización de RUP como la metodología a utilizar para el desarrollo de la solución pues las funcionalidades que se desarrollarán representan solo una parte de lo que se pretende lograr, donde las demás funcionalidades se desarrollarán en futuras versiones, además de ser una metodología donde los autores tienen experiencia pues se ha utilizado por estos en el desarrollo de otros softwares. A continuación se profundiza sobre las características que presenta RUP.

2.4.1 ¿Qué es RUP?

El Proceso Unificado de Desarrollo de Software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. (17)

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo.

RUP propone como sus principales elementos a:

1-Trabajadores: definen el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina y son los que realizan las actividades.

2-Actividades: Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

3-Artefactos: Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

4-Flujo de actividades: Una secuencia de actividades realizadas por trabajadores y que producen un resultado de valor observable.

El Proceso Unificado de Desarrollo de Software (RUP) se divide en cuatro fases y nueve flujos de trabajo para la mejor organización de las actividades a desarrollar.

Flujos de trabajo:

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.
- **Instalación:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Fases:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

- **Construcción:** Se obtiene un producto listo pata su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios release del producto que han pasado las pruebas. Se ponen los release a consideración de un subconjunto de usuarios.
- **Transición:** El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

2.4.2 Características de RUP.

En (17) se definen las principales características de RUP:

2.4.2.1 Iterativo e incremental.

La metodología RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Cada fase se subdivide en iteraciones. Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación, cuyo resultado es una versión del software.

2.4.2.2 Dirigido por casos de uso.

Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo pues los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

2.4.2.3 Centrado en la arquitectura.

La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. El modelo de arquitectura de un sistema se representa a través de vistas, donde se incluyen los elementos más significativos divididos en Vista Lógica, Vista de Procesos, Vista de Despliegue, Vista de Implementación y la Vista de Casos de Uso.

2.5 Herramientas de desarrollo.

2.5.1 Herramienta CASE. Visual Paradigm.

Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. En la actualidad existe un gran número de estas aplicaciones como son Rational Rose, Umbrello, Visual Paradigm las cuales se utilizan para la creación de artefactos durante el desarrollo de un software.

Se ha decidido utilizar Visual Paradigm porque es una herramienta CASE (Computer-Aided Software Engineering), gratuita utilizada para el modelado de aplicaciones, utiliza UML como lenguaje de modelado. Esta herramienta visual permite construir la aplicación con mayor rapidez, mayor exactitud, mejor trabajo en equipo, aumenta además las expectativas mediante la interfaz gráfica. Facilita la interoperabilidad con otras herramientas CASE, y con la mayoría de IDEs (Ambiente de Desarrollo Integrado), así mismo permite la integración de todos los componentes. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos y permite controles de versiones.

2.5.2 Entorno de desarrollo. NetBeans_IDE 6.0.

Un entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.(18)

Los dos entornos de desarrollo más potentes que existen en la actualidad para la programación en java son Eclipse y NetBeans. Ambos son gratuitos y poseen funcionalidades comunes que permiten a los programadores realizar cualquier operación. Por su parte Eclipse posee una serie de plugins con los cuales brinda una mayor funcionalidad al entorno de desarrollo. Este IDE es de gran utilidad para el desarrollo de aplicaciones web, lo cual se hace más sencillo y cómodo con la utilización de los antes mencionados plugins.

En el desarrollo de la aplicación se decidió utilizar NetBeans pues es superior al Eclipse en cuanto al entorno gráfico y la interfaz de usuario, dos aspectos fundamentales a tener en cuenta en el producto a desarrollar.

El NetBeans IDE es un entorno de desarrollo, una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

El enfoque de NetBeans IDE 6.0 es la productividad del desarrollador con un editor más inteligente y rápido, y la integración de todos los productos NetBeans en un solo IDE. El IDE en cuestión se ejecuta en muchas plataformas incluyendo Windows, Linux, Mac OS X y Solaris; y posee un proceso simplificado de instalación que permite instalar y configurar fácilmente el IDE para satisfacer exactamente varias necesidades.(19)

2.6 Tecnología Java.

C++ y Java son los dos lenguajes más populares y robustos de programación orientada a objetos. Ambos presentan características similares, pero se diferencian en una serie de elementos que hace que los programadores se inclinen hacia uno de ellos según el tipo de aplicación que se desee desarrollar. C++ es mucho más rápido que Java pues este genera un código máquina. Aunque existen muchas librerías en la red para C++, no son estándar del lenguaje, y algunas son de pago. Una de las principales ventajas de Java sobre C++ es la eliminación de punteros. Por otra parte C++ no es multiplataforma, para lograr aplicaciones que se ejecuten en varios sistemas operativos se requiere de cierto esfuerzo.

Según estas ventajas se decidió la utilización del lenguaje Java para el desarrollo de la aplicación, por ser de vital importancia el tema de la portabilidad y el desarrollo del lenguaje, permitiendo de esta manera el uso de la aplicación en cualquier sistema operativo y de las mejoras potenciales que se llevan a cabo en las librerías y en el lenguaje base, aunque estas últimas son muy inusuales.

2.6.1 Lenguaje de programación Java.

El lenguaje de programación Java es un lenguaje de programación de alto nivel y orientado a objeto el cual se basa en los lenguajes C y C++, pero es más fácil de utilizar y elimina herramientas de bajo nivel como la manipulación de punteros.

Según (20) el lenguaje Java posee como características principales:

Lenguaje simple.

Java posee una curva de aprendizaje muy rápida y las personas que hayan trabajado con C++ C les resultará muy fácil su familiarización con el lenguaje.

Orientado a Objetos.

Java es un lenguaje que fue creado desde sus inicios como un lenguaje orientado a objeto, utilizando clases, las cuales poseen datos y funciones encargadas de manejar estos datos.

Distribuido

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez.

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador.
Por otra parte, es interpretado, pues los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real.

Robusto.

Java proporciona un gran número de comprobaciones en compilación y en tiempo de ejecución, por lo que fue creado para crear aplicaciones altamente confiables.

Indiferente a la arquitectura

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecode: un formato intermedio indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

Portable

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

2.6.2 La Plataforma Java.

Con plataforma se refiere al ambiente de hardware y software en donde el programa se ejecuta, por ejemplo, plataformas como Windows, Solaris y Linux. En casi todos los casos las plataformas son descritas como la combinación de un sistema operativo y el hardware. La plataforma Java se diferencia de estas, pues es solo de software y se ejecuta sobre las otras plataformas de hardware. La plataforma Java está compuesta por la Máquina Virtual de Java (JVM) y las API de Java.

La JVM es una de las piezas fundamentales de la plataforma Java. Básicamente se sitúa en un nivel superior al Hardware del sistema sobre el que se pretende ejecutar la aplicación, y este actúa como un puente que entiende tanto el bytecode, como el sistema sobre el que se pretende ejecutar. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una máquina virtual Java en concreto, siendo ésta la que en última instancia convierte de código bytecode a código nativo del dispositivo final.

El API Java es una Interfaz de Programación de Aplicaciones (API: por sus siglas en ingles) provista por los creadores del lenguaje Java, y que da a los programadores los medios para desarrollar aplicaciones Java.

Como el lenguaje Java es un Lenguaje Orientado a Objetos, la API de Java provee de un conjunto de clases utilitarias para efectuar toda clase de tareas necesarias dentro de un programa. La API Java está organizada en paquetes lógicos, donde cada paquete contiene un conjunto de clases relacionadas semánticamente.

La plataforma seleccionada para el desarrollo de la aplicación fue J2SE[4], por sus características para aplicaciones de escritorio.

2.7 Conclusiones.

Al concluir el presente capítulo se tiene un conocimiento de las distintas tecnologías y herramientas que se utilizaron para el desarrollo del sistema y de sus características y ventajas sobre otras existentes.

Es importante el análisis de estas tecnologías porque no solo se utilizan porque sean las más actuales o las más utilizadas sino que se estudian sus beneficios y se comprende mejor el funcionamiento de las mismas, lo que hace más fácil su utilización en el desarrollo del futuro sistema.

[4] J2SE: se refiere a Java 2 Edición Estándar (por sus siglas en inglés).

CAPÍTULO 3. Presentación de la solución propuesta.

3.1 Introducción.

En el siguiente capítulo se analizará la solución propuesta para lo cual se analizará el entorno donde trabajará el sistema, así como sus conceptos y eventos principales, se listan los requisitos funcionales y no funcionales del software a obtener y se muestra el diagrama de casos de uso del sistema y se describen dichos casos de uso para su mejor comprensión.

3.2 Entorno donde trabajará el sistema.

Un modelo de Dominio se representa cuando no se tienen procesos bien definidos. El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema. Representa conceptos del mundo real, no de los componentes de software. Para la construcción de un modelo de dominio se extraen los conceptos y eventos principales del entorno y se relacionan en un diagrama de clases denominado **Diagrama de Clases del Modelo de Dominio.**

3.2.1 Conceptos principales del entorno.

Simulador de procesos.

Simulador de Proceso es un sistema independiente que se utiliza para señalar el montaje en una computadora de los modelos del proceso que funcionan en dicho montaje y se convierte en un instrumento para experimentar y aprender de las consecuencias de las decisiones futuras. El simulador es un software desarrollado en el Polo Científico de Simulación que se desarrolla en la Facultad 9 de la Universidad de las Ciencias Informáticas.

Gráfica.

Una **gráfica** es la representación de datos, generalmente numéricos, mediante líneas, superficies o símbolos, para ver la relación que esos datos guardan entre sí.

3.2.2 Eventos principales del entorno.

Simulación.

Conjunto de actividades que se desarrollan para llevar a cabo la representación de un evento de la vida diaria mediante un modelo. Un proceso de simulación esta compuesto por las fases de desarrollo del modelo, ejecución del modelo y análisis de las salidas del modelo.

Graficación.

La graficación es un evento mediante el cual los datos deseados van a ser traducidos a un tipo de gráfica, la cual mostrará de forma más elocuente la relación que existe entre los mismos. Estas gráficas se pueden representar o no en un sistema de coordenadas en el plano en dependencia del tipo que sean.

3.2.3 Diagrama de Clases del Modelo de Dominio.

Diagrama de clases del modelo de dominio

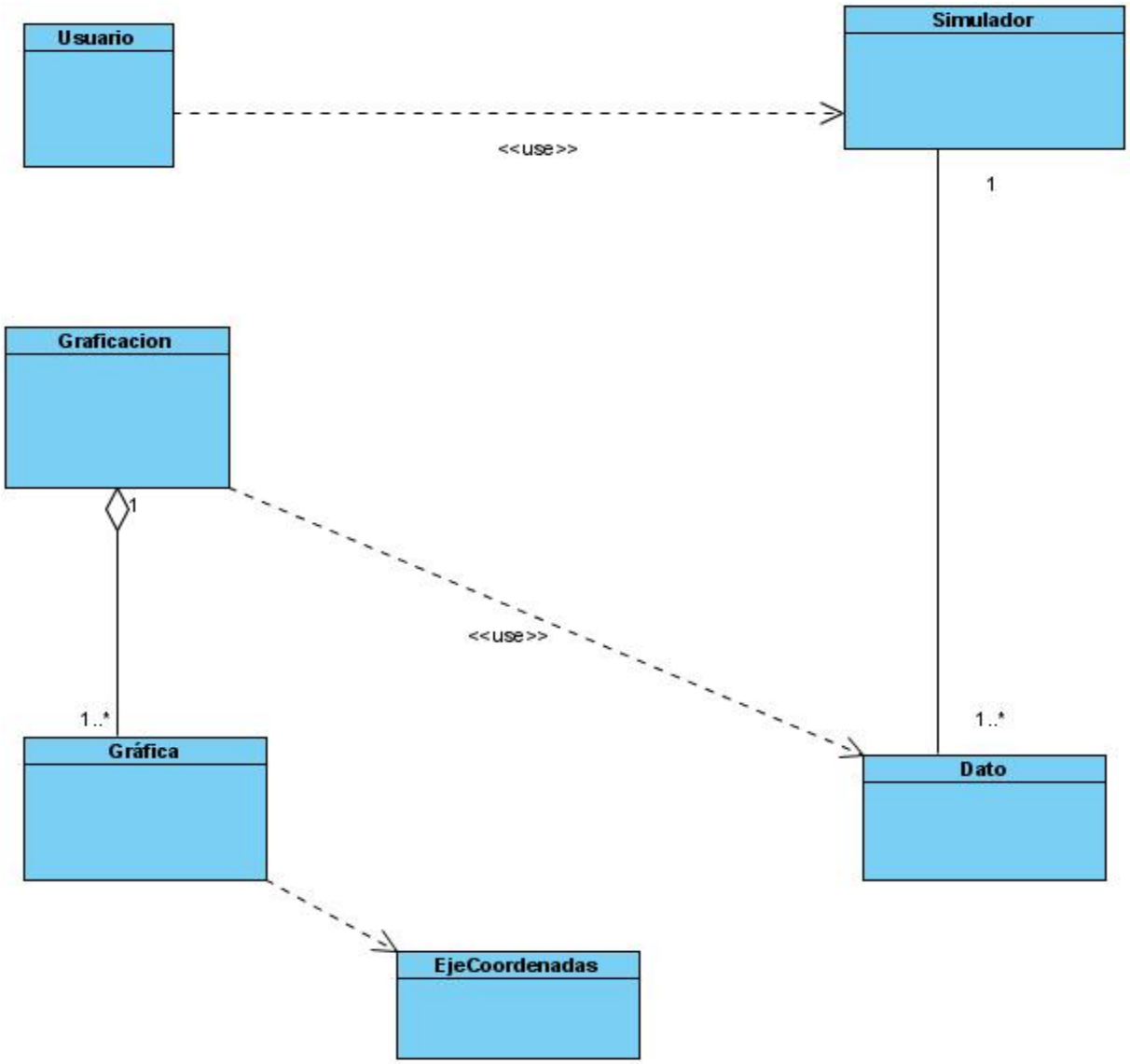


Figura 10 Diagrama de clases del modelo de dominio.

3.2.4 Glosario de Términos del Dominio.

Polo científico.

Es la agrupación de varios factores como el humano y tecnológico con el objetivo de investigar, obtener conocimientos y desarrollar productos en cierta rama, que sean de utilidad para la sociedad o para parte de ella.

Sistema de coordenadas en el plano.

Compuestas por dos ejes(x e y) que dividen el plano en cuatro cuadrantes. A los ejes también se les llama abscisa(x) y ordenadas (y).Las coordenadas varían de positivo a negativo según el cuadrante en que se encuentren.

3.3 Requisitos Funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, los mismos se muestran a continuación:

RF1 Gestionar los datos que serán utilizados en la graficación.

RF2 Graficar datos en el plano.

2.1 En forma de barra.

2.2 En forma de líneas.

2.3 En forma de pastel.

RF3 Convertir de un tipo de gráfica a otro.

3.1 Convertir a gráfica de barra.

3.2 Convertir a gráfica de línea.

3.3 Convertir a gráfica de pastel.

RF4 Construir ejes coordenados.

RF5 Guardar imagen de una gráfica en un archivo.

3.4 Requisitos No Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. En el producto que se desarrollará se definirán los siguientes requisitos no funcionales:

Apariencia o interfaz externa.

- El sistema deberá tener una interfaz externa amigable, que sea sencilla y fácil de entender por el usuario para así evitar que al usuario se le haga engorroso el trabajo con la misma.

Portabilidad.

-El sistema funcionará en cualquier sistema operativo, para lo cual será necesario contar de la Máquina Virtual de Java para el sistema operativo en el cual se encuentre el sistema.

Software.

-No hay restricciones en cuanto al sistema operativo a instalar puesto que la aplicación será multiplataforma.

- La Máquina Virtual de Java tiene que estar instalada.

Hardware.

-Se debe contar con 256 MB de memoria RAM como mínimo, aunque lo ideal sería 512 MB.

Requerimientos en el diseño y la implementación.

- Se utiliza el lenguaje de programación Java.

- Se utiliza NetBeans como herramienta de desarrollo.

- Se utiliza Visual Paradigm para el análisis y diseño de la aplicación.

Usabilidad.

- El sistema debe ser usado por cualquier persona que desee representar en forma de gráfica los datos que se obtengan en los procesos de simulación. La interfaz del usuario deberá ser tan familiar como sea posible a los usuarios y se debe poseer una ayuda para el mejor uso.

3.5 Descripción del Sistema Propuesto.

3.5.1 Descripción de los actores.

Un actor del sistema es una persona, sistema o entidad que interactúa con el sistema y que cumplen un rol determinado. Un actor no forma parte del sistema. En la siguiente tabla se muestra el actor del sistema **(Usuario)** y la justificación.

Actores	Justificación
Usuario.	Se encarga de manejar los datos que serán objeto de la graficación. Es la persona que interactúa con la aplicación para construir los ejes coordenados que se necesitan para la graficación. Es el responsable de la graficación de los datos en el plano en las distintas formas. Es el responsable de guardar la imagen de una gráfica en un archivo.

Tabla 2 Actores del Sistema

3.5.2 Casos de Uso del Sistema.

Un caso de uso representa la forma de capturar requisitos de un nuevo sistema. Estos casos de uso se representan mediante un diagrama de casos de uso, estos muestran la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo. A continuación se representa el diagrama de casos de uso del sistema a desarrollar.

DIAGRAMA DE CASOS DE USO

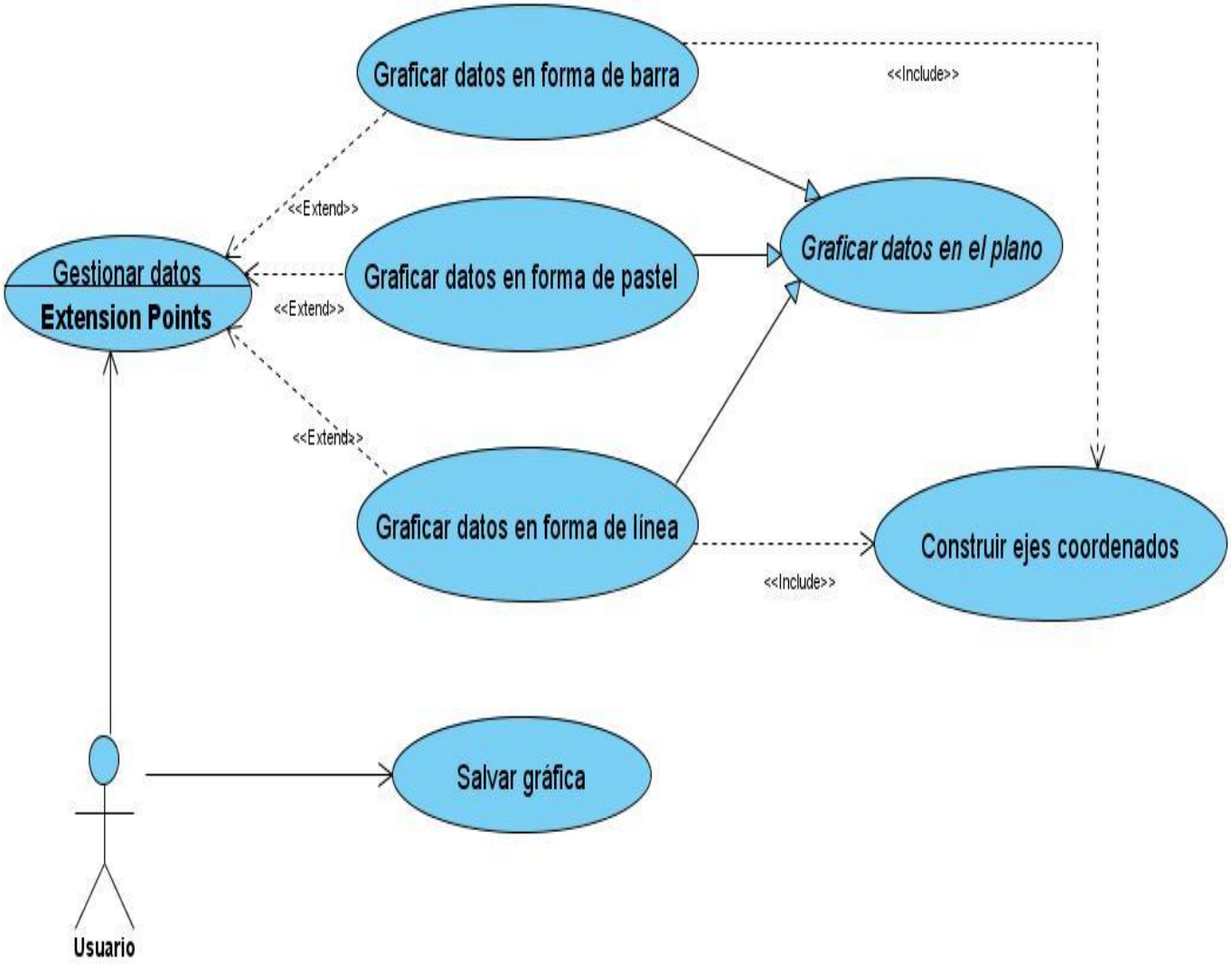


Figura 11 Diagrama de casos de uso del sistema

3.5.2.1 Descripción textual de los casos de uso del sistema.

3.5.2.1.1 Caso de Uso Gestionar datos.

Caso de uso	Gestionar datos.
Actores	Usuario (Inicia).
Propósito	Ingresar los datos necesarios para la graficación de las distintas gráficas que puedan ser construidas posteriormente
Resumen	El caso de uso se inicia cuando el usuario solicita ingresar datos para su posterior graficación .Se ingresan los parámetros a graficar y sus respectivos datos.
Referencias	RF 1
Precondiciones	Se tiene un usuario con la necesidad de graficar datos.

Curso Normal de los eventos.	
Acción del actor.	**Respuesta del sistema.**
1. Solicita la opción de ingresar datos.	2. El sistema brinda la opción de adicionar parámetros o series.
3. El usuario selecciona la opción de adicionar los parámetros a graficar.	4. El sistema permite entrar los nombres de las categorías que se graficarán.
5. El usuario solicita entrar respecto a que se van a graficar los parámetros (series).	6. El sistema permite ingresar los elementos del eje horizontal, respecto a las cuales se graficarán las categorías insertadas anteriormente.
7. El usuario entra los datos correspondientes. 8.El usuario decide : a) Graficar en forma de barra (Ver **CU Graficar datos en forma de barra**). b) Graficar en forma de línea (**Ver CU**	

Graficar datos en forma de línea). c) Graficar en forma de pastel (**Ver CU Graficar datos en forma de pastel**).	
Curso alterno	
Línea 4 y 6. Si ocurre alguna anomalía al entrar un nombre se muestra un mensaje.	
Poscondiciones.	Se tienen los datos a graficar.

Tabla 3 Manejar datos.

3.5.2.1.2 Caso de Uso Graficar datos en forma de barra.

Caso de uso	**Graficar datos en forma de barra.**
Actores	Usuario (Inicia).
Propósito	Graficar datos en forma de barra para la mejor comprensión de los mismos.
Resumen	El caso de uso se inicia cuando el actor decide graficar los datos en este tipo de gráfica, para lo cual es necesario construir los ejes coordenados correspondientes.
Referencias	RF 2.1 ,RF 3.1
Precondiciones	Es necesario tener los datos que van a ser objeto de la graficación.
Curso Normal de los eventos.	
Acción del actor.	**Respuesta del sistema.**
1. El usuario solicita graficar en forma de barra. (Se invoca el CU **Construir ejes coordenados**). 2. El usuario selecciona la opción de aceptar.	3. El sistema muestra la gráfica de barra correspondiente. Termina el caso de uso.
Curso alterno.	
Línea 2. Si se selecciona cancelar termina el caso de uso.	
Poscondiciones.	Se obtiene la gráfica correspondiente.

Prioridad	Crítico

Tabla 4 Graficar datos en forma de barra.

3.5.2.1.3 Caso de Uso Graficar datos en forma de línea.

Caso de uso	**Graficar datos en forma de línea.**
Actores	Usuario (Inicia).
Propósito	Graficar datos en forma de línea para la mejor comprensión de los mismos.
Resumen	El caso de uso se inicia cuando el actor decide graficar los datos en este tipo de gráfica, para lo cual es necesario construir los ejes coordenados correspondientes.
Referencias	RF 2.2, RF 3.2
Precondiciones	Es necesario tener los datos que van a ser objeto de la graficación.
Curso Normal de los eventos.	
Acción del actor.	**Respuesta del sistema.**
1. El usuario solicita graficar en forma de línea. (Se invoca el **CU Construir ejes coordenados**). 2. El usuario selecciona la opción de aceptar.	3. El sistema muestra la gráfica de línea correspondiente. Termina el caso de uso.
Curso alterno.	
Línea 2. Si se selecciona la opción de cancelar se termina el caso de uso.	
Poscondiciones.	Se obtiene la gráfica correspondiente.
Prioridad	Crítico.

Tabla 5 Graficar datos en forma de línea.

3.5.2.1.4 Caso de Uso Graficar datos en forma de pastel.

Caso de uso	**Graficar datos en forma de pastel.**
Actores	Usuario (Inicia).
Propósito	Graficar datos en forma de pastel para la mejor comprensión de los mismos.
Resumen	El caso de uso se inicia cuando el actor decide graficar los datos en este tipo de gráfica, para lo cual es necesario el titulo del gráfico en cuestión.
Referencias	RF 2.3, RF 3.3
Precondiciones	Es necesario tener los datos que van a ser objeto de la graficación.
Curso Normal de los eventos.	
Acción del actor.	**Respuesta del sistema.**
1. El usuario selecciona la opción de graficar en forma de pastel.	2. El sistema da la posibilidad de seleccionar el parámetro que se va a graficar.
3. El usuario escoge el parámetro.	4. El sistema muestra la gráfica de pastel correspondiente.
Curso alterno.	
Línea 4. Si no se escogió el parámetro se muestra un mensaje para que lo ingrese.	
Poscondiciones.	Se obtiene la gráfica correspondiente.
Prioridad	Crítico.

Tabla 6 Graficar datos en forma e pastel

3.5.2.1.5 Caso de uso Construir ejes coordenados.

Caso de uso	**Construir ejes coordenados.**
Actores	Usuario.
Propósito	Construir ejes coordenados para la graficación de los datos

Resumen	El caso de uso inicia cuando el usuario escoge que desea graficar sus datos en forma de barra o línea, se ingresan los nombres de los ejes y el título del gráfico.
Referencias	RF 4.
Precondiciones	El usuario ha seleccionado que desea graficar los datos en forma de barra o línea.
Curso Normal de los eventos.	
Acción del actor.	**Respuesta del sistema.**
	1. El sistema solicita ingresar el nombre de los ejes coordenados y el título del gráfico.
2. El usuario ingresa los nombres correspondientes a cada campo.	
Curso alterno.	
Línea 2.Si se deja en blanco algún campo el sistema muestra un mensaje y gráfica.	
Poscondiciones.	Se obtienen los ejes coordenados para la graficación de los datos.
Prioridad	Crítico.

Tabla 7 Construir ejes coordenados

3.5.2.1.6 Caso de uso Salvar gráfica.

Caso de uso	**Salvar gráfica**
Actores	Usuario.
Propósito	Salvar la imagen de una gráfica en un archivo.
Resumen	El caso de uso inicia cuando el usuario escoge que desea graficar sus datos en forma de barra o línea, se ingresan los nombres de los ejes y el título del gráfico.
Referencias	RF 5.
Precondiciones	Tiene que estar graficada la gráfica que se desea salvar.

Curso Normal de los eventos.	
Acción del actor.	**Respuesta del sistema.**
1. El usuario solicita la opción de salvar la gráfica.	2. El sistema solicita ingresar el nombre del fichero y su ubicación.
3. El usuario ingresa el nombre y le da la ubicación.	4. El sistema guarda la imagen de la gráfica. Termina el caso de uso.
Curso alterno.	
Línea 3.Si se ingresa un nombre que ya existe en esa ubicación se muestra un mensaje.	
Poscondiciones.	Se obtiene un fichero con la imagen de la gráfica
Prioridad	Secundario.

Tabla 8 Salvar gráfica

3.6 Conclusiones.

En el presente capítulo se abordaron los aspectos más importantes de la propuesta de solución del sistema .Al concluir el mismo se tiene un conocimiento de lo que hará el sistema sin entrar en los detalles de cómo lo hará.

Los autores consideran que el desarrollo de este capítulo constituye un paso esencial en el desarrollo del sistema, pues se aborda desde el entorno donde trabajará el sistema (de gran importancia para el desarrollo del mismo) hasta el diagrama de casos de uso del sistema que es la representación gráfica de los casos de uso, los cuales responden a los requisitos que debe cumplir el sistema. Con el desarrollo del mismo se crea una base sólida para guiar los esfuerzos del sistema basándose en los casos de uso identificados.

CAPÍTULO 4. Construcción de la solución propuesta.

4.1Introducción.

En el actual capítulo se exponen los principales elementos que sustentan la construcción de la propuesta de solución. En el mismo se mostrarán varios artefactos que tienen como objetivos mostrar las principales actividades de los flujos de trabajo de diseño e implementación del sistema propuesto como solución, tales como el modelo de diseño e implementación.

4.2 Patrones GRASP.

Los patrones son parejas de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. En el diseño del sistema se pusieron en práctica el uso de algunos patrones de software para la asignación general de responsabilidades (GRASP).

A continuación se detallan algunos de los patrones GRASP que se pusieron en práctica en el diseño de la aplicación.

4.2.1 Patrón Controlador.

El patrón Controlador asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente el sistema real, la organización o empresa.

4.2.2 Patrón Creador.

El patrón Creador se plantea como problema ¿Quién debería ser responsable de crear una nueva instancia de alguna clase? Y para ello la solución es asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

1. B agrega los objetos A.
2. B contiene los objetos A.
3. B registra las instancias de los objetos A.
4. B utiliza específicamente los objetos A.

4.2.3 Patrón Polimorfismo.

El patrón Polimorfismo plantea como problema ¿Cómo manejar las alternativas basadas en el tipo? y como solución la asignación de operaciones polimórficas a los tipos en los que el comportamiento muestre variaciones.

4.2.4 Patrón Experto.

Este patrón es el principio básico de asignación de responsabilidades. El mismo indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Se plantea como problema ¿Cuál es el principio general para asignar responsabilidades a los objetos?

Solución: Asignar una responsabilidad al experto en información.

Es importante la utilización de este patrón pues mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener.

4.3 Patrón arquitectónico.

El sistema que se desarrolló al no tener la necesidad de poseer un mecanismo para el almacenamiento de datos, pues la información de los datos con que se trabaja no es necesario que perdure, se dividirá en dos capas lógicas (la encargada de la presentación y la de lógica del negocio). En la capa de presentación se tienen todas las interfaces de usuario necesarias para que los usuarios puedan utilizar la aplicación. Esta capa se comunica con la de negocio. La capa de negocio, se denomina de esta manera pues se establecen todas las reglas que deben cumplirse y se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados.

4.4 Diagrama de Clases del Diseño.

Un Diagrama de Clases de Diseño es un diagrama estático que muestra la especificación para las clases software de una aplicación (15). Incluye la siguiente información:

• Clases, asociaciones y atributos.
• Interfaces, con sus operaciones y constantes.
• Métodos.
• Navegabilidad.
• Dependencias.

A diferencia del Modelo Conceptual, un Diagrama de Clases de Diseño muestra definiciones de entidades software más que conceptos del mundo real. El diagrama de clases del diseño es el diagrama más importante de la fase de diseño de un software, el mismo brinda una visión general para comenzar con la implementación del producto.

Para la creación del diagrama de clases del diseño primeramente se identifican las entidades software necesarias para desarrollar el sistema que se pretende realizar, se definen las relaciones existentes entre ellas y los atributos y métodos según la necesidad del sistema. Para la definición de forma más sencilla de los métodos en cada clase es muy factible el apoyo en los diagramas de interacción (colaboración o secuencia).

Los diagramas de interacción muestran el paso de mensajes necesarios, entre las instancias de los objetos, para desarrollar un escenario de un caso de uso. En el desarrollo del actual software se utilizaron los de colaboración por su excepcional expresividad, su capacidad de comunicar más información contextual y su economía de espacio. Un mensaje dirigido a una instancia de una clase significa que dicha clase debe definir el método con el nombre de dicho mensaje. En la práctica los diagramas de clase del diseño se pueden desarrollar al mismo tiempo que los de interacción.

Teniendo en cuenta las operaciones que debe poseer el sistema a desarrollar se realizaron los siguientes diagramas de colaboración.

Diagrama colaboración CU Graficar datos en forma de barra

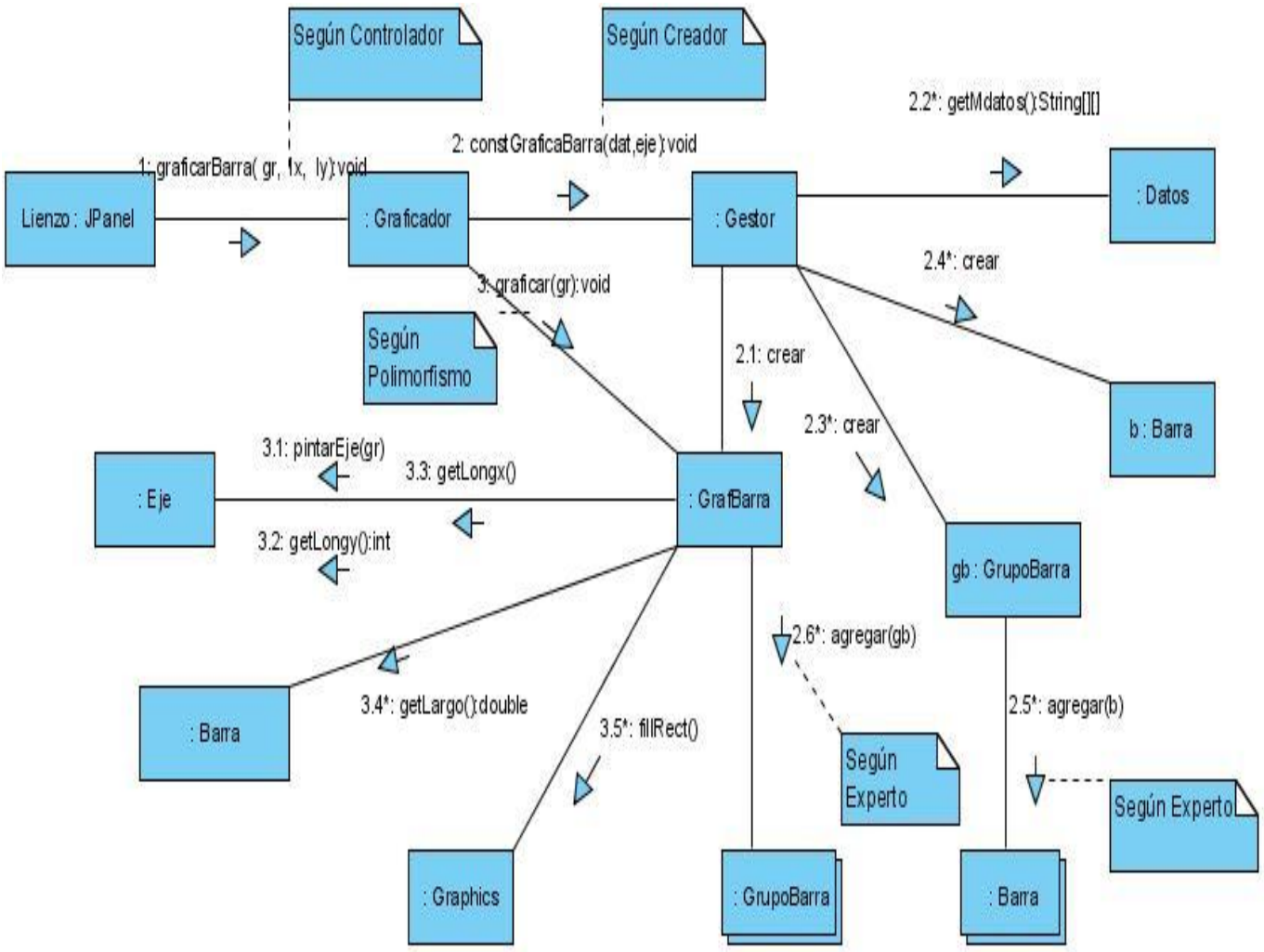


Figura 12 Diagrama de colaboración CU Graficar datos en forma de barra.

Diagrama colaboración CU Graficar datos en forma de línea

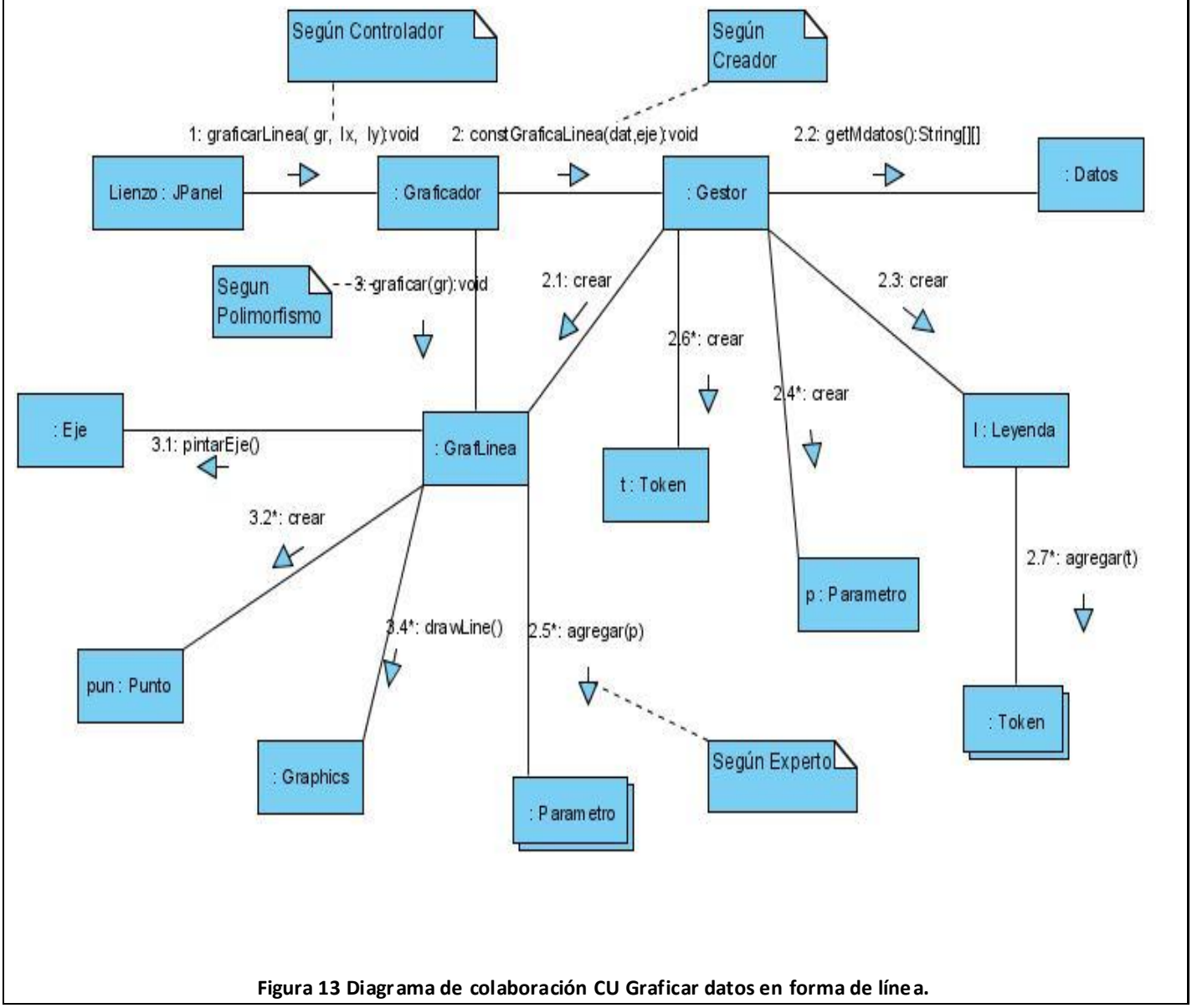


Figura 13 Diagrama de colaboración CU Graficar datos en forma de línea.

Diagrama colaboración CU Graficar datos en forma de pastel

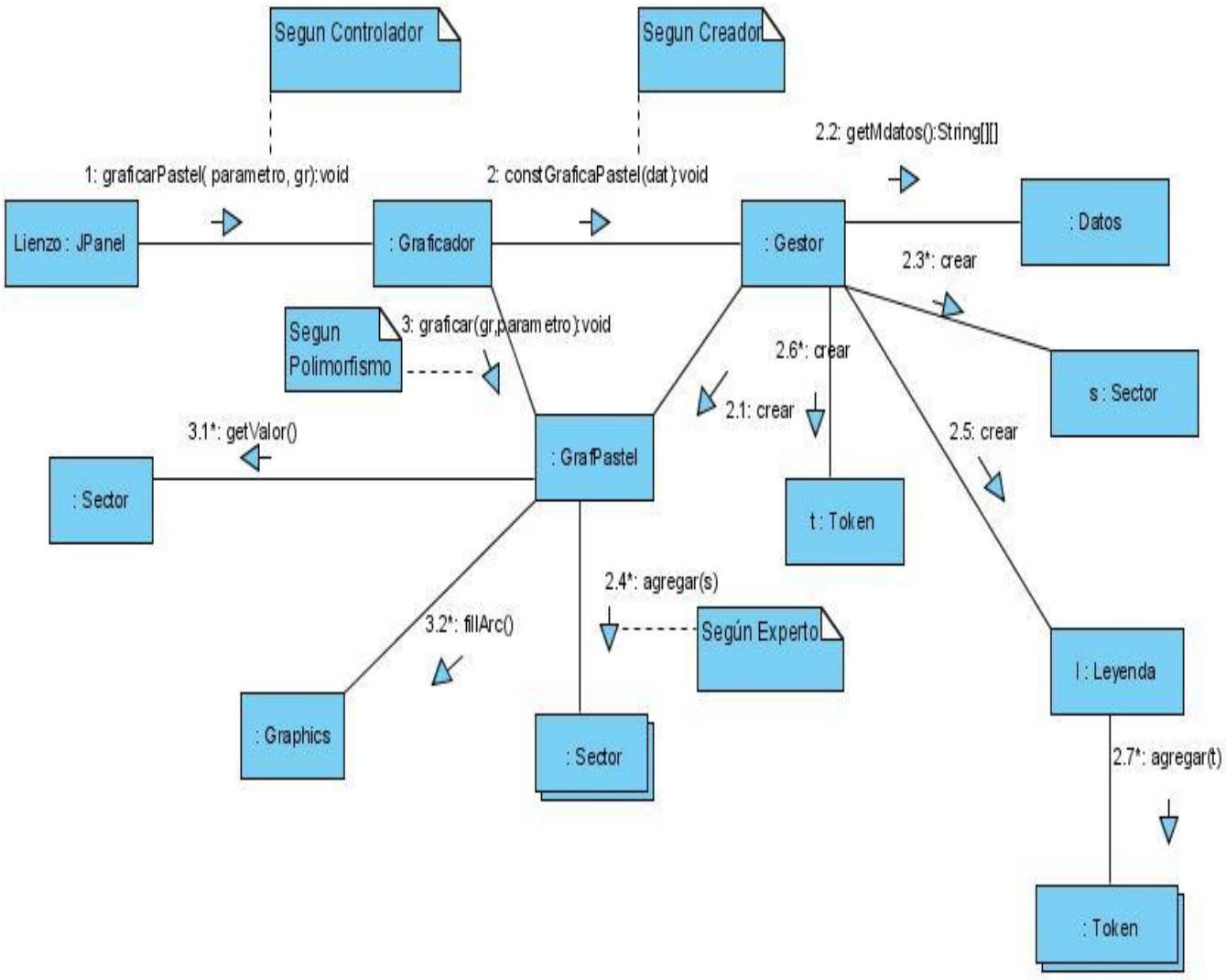


Figura 14 Diagrama de colaboración CU Graficar datos en forma de pastel.

Diagrama colaboración CU Construir ejes coordenados

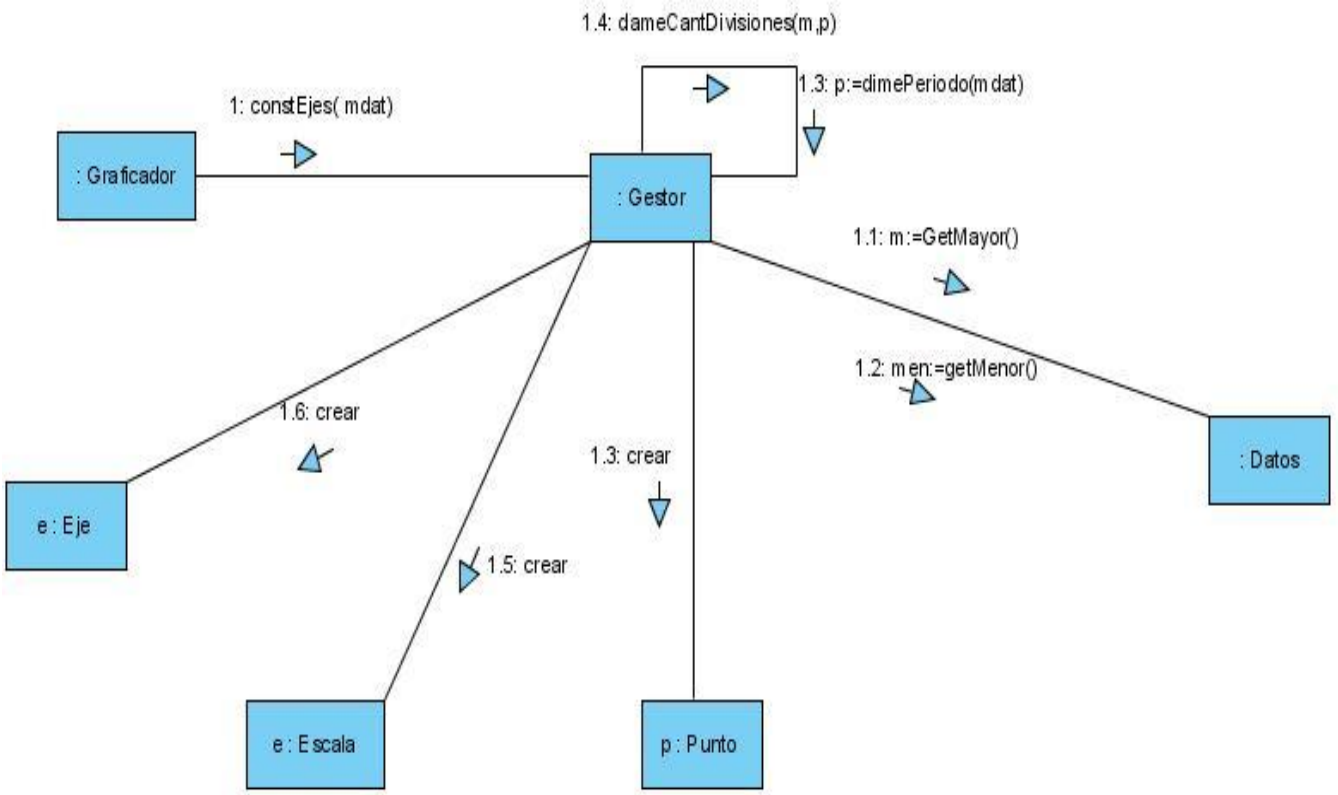


Figura 15 Diagrama de colaboración CU Construir ejes coordenados.

A continuación con los diagramas de colaboración se identifican las clases que participan en la solución del software y se agregan los nombres de los métodos para cada clase, así como la información sobre los tipos a los atributos y métodos.

Es importante destacar que en el sistema a desarrollar al igual que en la mayoría de los softwares el diagrama de clases del diseño se actualiza, pues aparecen nuevos elementos, a medida que avanza el desarrollo del sistema hasta obtener la versión final del mismo que se corresponde con el diseño final del software.

Seguidamente se muestra el diagrama de clases del diseño resultante correspondiente al software de graficación que se desarrolló.

Diagrama de clases del diseño

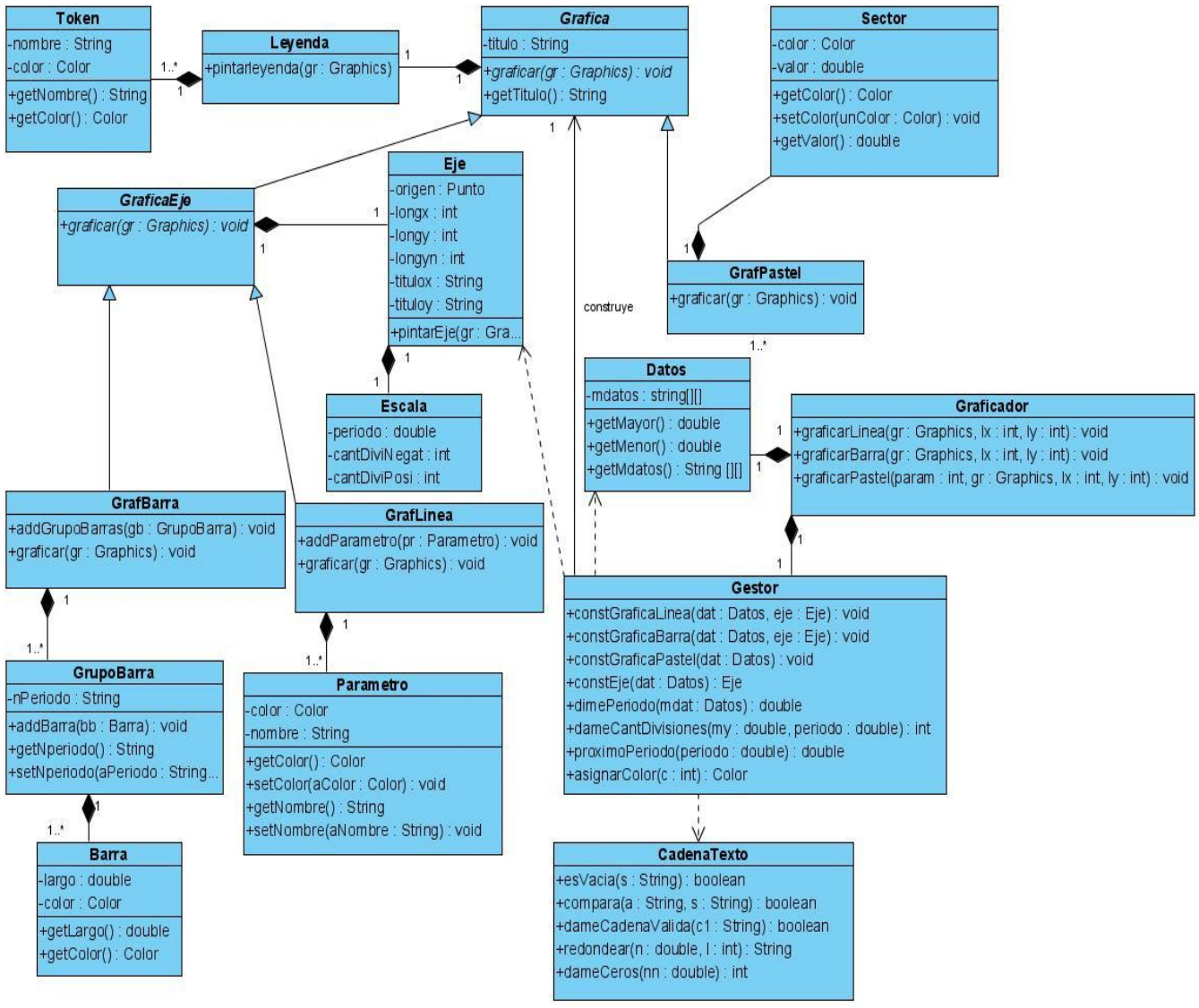


Figura 16 Diagrama de clases del diseño.

4.5 Interfaz de usuario.

La aplicación constará de dos formas fundamentales (ver anexos), las cuales se encargarán de brindarles a los usuarios la posibilidad de manejar los datos que serán objeto de la graficación y de la representación de las gráficas en cuestión respectivamente. Además de estas dos formas existirán otras de menor responsabilidad, para la introducción de los parámetros y títulos.

La forma encargada de manejar los datos que serán graficados se comunica con la segunda forma y esta es la que interactúa, a través de una instancia de la clase Lienzo, con la capa de negocio mandándole los mensajes necesarios para llevar a cabo las operaciones del sistema, lo cual responde al siguiente principio de diseño:

1. La capa de presentación no debe tener responsabilidades lógicas de dominio, tan sólo debería encargarse de las tareas de presentación (interfaz).
2. La capa de presentación debe enviarle a la capa del dominio las peticiones de las tareas orientadas al dominio, que es la que se ocupa de ellas.

4.6 Modelo de implementación.

Un modelo de implementación describe como los elementos del diseño, como son las clases, se implementan en términos de componentes, como fichero de código fuente, librerías, ejecutables. El modelo de implementación describe también como se organizan los componentes de acuerdo al lenguaje de programación utilizado y al entorno de implementación y como dependen los componentes entre sí. Un componente es el empaquetamiento físico de un elemento del diseño, como lo son las clases en el modelo de diseño.

Según el lenguaje de programación utilizado, los componentes que se generan tienen sus características específicas pues como se ha expresado anteriormente ellos no son más que el empaquetamiento físico de un elemento.

Particularmente en el lenguaje Java existen varios estereotipos para diferenciar los distintos tipos de componentes en un diagrama de componentes. Un archivo de clases (.java) es representado por el estándar UML <<file>> y al crear el componente de dicha clase se estereotipa con <<JavaClassFile>>. Típicamente un archivo JAR contiene los archivos de clase y recursos auxiliares

asociados con programas y aplicaciones. Estos archivos son representados en UML mediante un paquete con el estereotipo <<JavaArchiveFile>>.

Teniendo en cuenta esta forma de estereotipar los componentes en el lenguaje JAVA y que cada clase se convierte en un componente de extensión .java, el diagrama de componentes del sistema es el siguiente.

Diagrama de componentes

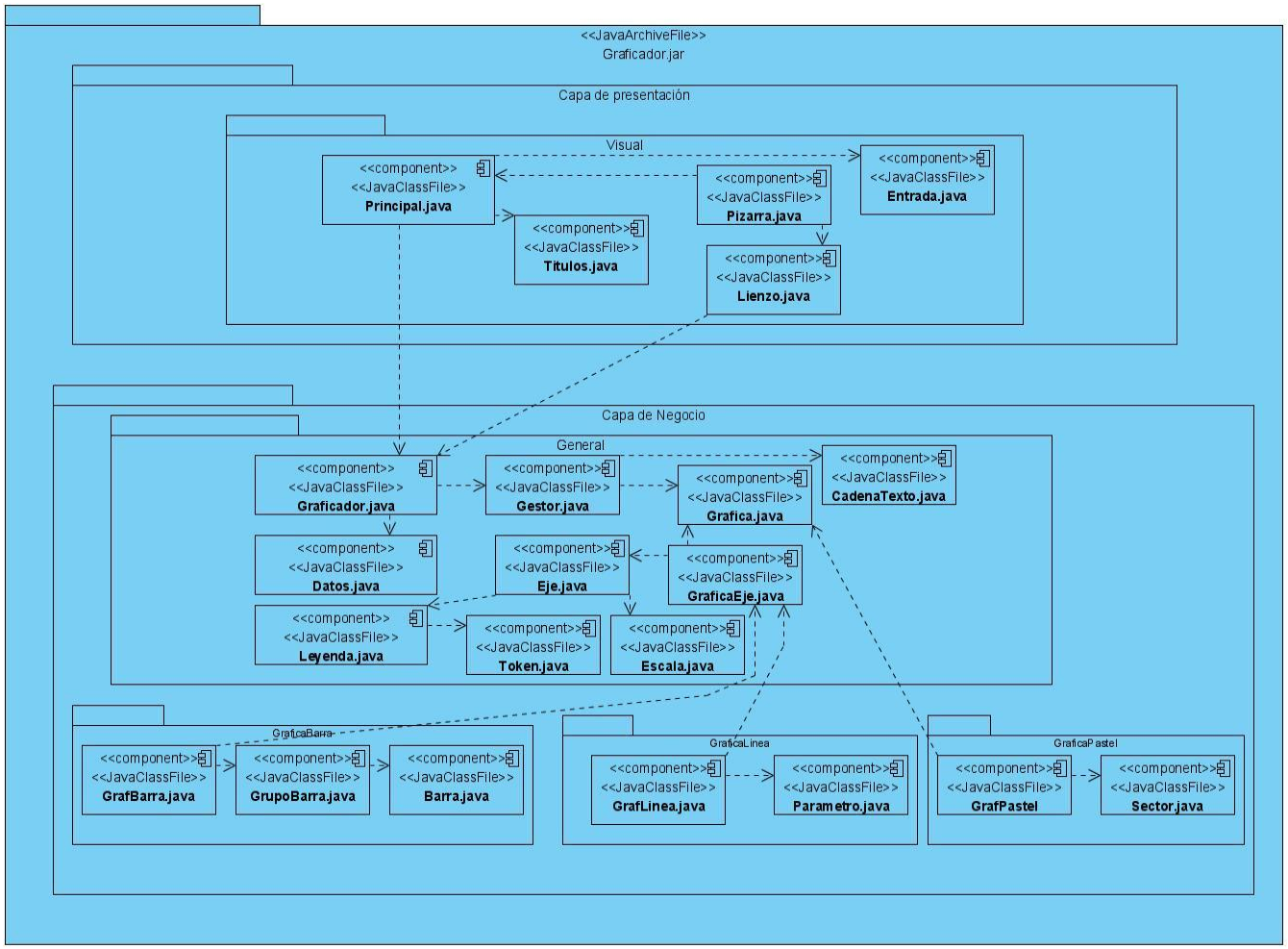


Figura 17 Diagrama de componentes.

Como se puede observar en el diagrama de componentes la aplicación esta dividida en dos capas y en el mismo se muestran los componentes divididos en las mismas y las relaciones entre ellos.

4.7 Aspectos generales de la implementación.

4.7.1 La clase Graphics en el desarrollo de la solución.

En el lenguaje Java es posible realizar operaciones gráficas a través de la clase Graphics, la cual pertenece al paquete java.awt .Esta clase tiene como tareas fundamentales la creación de un contexto gráfico y la de proporcionar los métodos necesarios para poder dibujar en la pantalla. El contexto gráfico incluye una serie de elementos tales como:

1. El objeto Component sobre el que se va a dibujar.
2. Un origen de coordenadas. Se especifican en pixeles y comienzan en la esquina superior izquierda (0,0). No obstante, recordar que las coordenadas son relativas a la ventana en la que se este trabajando.
3. Los colores del fondo y el primer plano .Por defecto son negro y blanco respectivamente.
4. La fuente de texto. Por defecto es la fuente Dialog, estilo plano y 12 pulgadas.

Es importante dibujar redefiniendo el método *paint()* para que el dibujo se mantenga al redimensionar la ventana, al minimizarla y luego maximizarla o pasarle otra ventana por encima .Para lograr este objetivo son de vital importancia el método *repaint()* y el *update(Graphics g)*.La llamada del *repaint()* se hace en el código y este avisa a la máquina virtual de que el componente debe ser repintado, cuando la máquina decide repintar dicho componente hace una llamada al método *update(Graphics g)* el cual borra todo en el componente y hace la llamada al *paint()*.

En el sistema que se desarrolló se creó una clase que hereda de un componente (JPanel) y se redefinió el método *paint ()*, logrando que lo dibujado en ese método se dibuje sobre dicho componente utilizando el Graphics pasado por parámetros. De esta manera se posibilitó el uso del repintado explicado anteriormente.

Existen otros métodos de la clase Graphics como los draw y fill los cuales se utilizan para el dibujado de las distintas figuras geométricas, siendo el draw el que dibuja el contorno de la figura solamente y el fill rellena la misma.

Para la graficación de los datos en forma de barra se utilizó el método *fillRect ()* el cual pinta un rectángulo rellenándolo del color que en ese momento tenga especificado el Graphics. Para la gráfica de línea se utilizó el *drawLine ()* el cual pinta un recta entre dos puntos del sistema de coordenadas y para la de pastel el *drawArc (),* para la representación de arcos de circunferencias definidos por un rectángulo. La construcción del eje para la representación de las gráficas y de la leyenda de las mismas se utilizó *drawLine ()* y *fillRect ()* respectivamente, además del *drawString ()* para especificar los nombres dentro del Graphics.

4.7.2 Tratamiento de errores.

La recuperación mejorada de errores es una de las formas más poderosas de incrementar la fortaleza del código. La recuperación de errores es fundamental en todos los programas que se escriban, pero es especialmente importante en Java, donde uno de los objetivos principales es crear componentes de programa para que otros los usen. Para crear un sistema robusto, cada componente debe ser robusto.

Los objetivos del manejo de excepciones en Java son simplificar la creación de programas grandes y seguros utilizando menos código de lo actualmente disponible, y con garantías de que la aplicación no tenga errores sin manejar.

Para garantizar la estabilidad y confiabilidad del sistema y prevenir errores por parte del usuario, se realizan validaciones siempre que se introduzcan datos .De esta forma se impide que el sistema pueda presentar comportamientos inesperados o falle al tratar de procesar datos incorrectos.

4.8 Pruebas del sistema.

Cuando se ha desarrollado un software las pruebas de software es el proceso que permiten verificar y revelar la calidad del mismo. Para determinar dicho nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones iniciales del sistema.

Una de las pruebas que se llevan a cabo son las pruebas de sistema, las cuales prueban al software funcionando como un todo, estas pruebas se llevan a cabo durante la fase de construcción. A través de estas pruebas cualquier producto de software puede ser probado de una de las siguientes formas:

1. Conociendo la funcionalidad específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.

2. Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que "todas las piezas encajen", o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

El 1er enfoque se denomina Prueba de Caja Negra y el 2do Prueba de Caja Blanca.

Prueba de caja negra: Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software.

Prueba de caja blanca: Se basa en el minucioso examen de los detalles de procedimientos. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y bucles para determinar si el estado real coincide con el esperado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar.

Un caso de prueba es un artefacto esencial durante el desarrollo de las pruebas ,los mismos consisten en un conjunto de entradas con datos de prueba, unas condiciones de ejecución, y unos resultados esperados cuyo propósito es identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los casos de la prueba son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto (casos de uso).

4.8.1 Casos de prueba.

El método que se propone para el diseñar los casos de prueba es el de pruebas de caja negra basada en la descripción de los casos de uso. El método consta de tres pasos fundamentales:

1. Para cada caso del uso, generar un sistema completo de escenarios.

Para cada caso de uso se crea una matriz denominada Matriz Parcial de Escenarios. Cada camino posible en un caso de uso (todas combinaciones entre el camino principal y los caminos alternativos)

se denomina escenario. La matriz cuenta con tres columnas en las cuales se representan el nombre del escenario, el flujo donde comienza y si contiene algún flujo alternativo.

2. Para cada escenario, identificar por lo menos un caso de prueba y las condiciones que hagan que se lleve a cabo.

Posteriormente se determinan las variables que determinan los escenarios de cada caso de uso y se representa en una matriz llamada Matriz de Casos de Prueba, la misma contiene además el comportamiento esperado del sistema.

3. Para cada caso de prueba, identificar los valores de los datos con los cuales se hará la prueba.

Como el objetivo final es realizar las pruebas con datos reales, el último paso es identificar los valores para cada variable en los distintos escenarios. En este paso a la matriz de casos de pruebas se le agrega la columna donde se especifica el resultado de la prueba.

Teniendo en cuenta los pasos descritos anteriormente se presentan algunos casos de prueba diseñados para probar las funcionalidades del software.

Id del escenario	**Escenario**	**Parámetros**	**Series**	**Datos**	**Respuesta del Sistema**	**Resultado de la prueba**
EC 1	Escenario1: Manejo exitoso de datos	El campo recibe un nombre real.	El campo recibe un nombre real.	Contiene los datos a graficar ,entrados correctamente	El sistema tendrá los datos necesarios para la graficación.	Se especifica que sucedió al realizar la prueba, pueden existir dos variantes: que la respuesta del sistema sea la esperada o que no lo sea.

EC 2	Escenario 2: Ausencia de datos.	Algún campo contiene los valores con alguna anomalía	El sistema emite un mensaje de anomalía.	Se especifica que sucedió al realizar la prueba, pueden existir dos variantes: que la respuesta del sistema sea la esperada o que no lo sea.

Tabla 9 Caso de prueba CU Manejar datos

Id del escenario	Escenario	Datos	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario1: Graficación exitosa de datos	Se tienen el conjunto de datos que serán objeto de la graficación.	Se mostrará la gráfica de barra correspondiente a los datos.	Se especifica que sucedió al realizar la prueba, pueden existir dos variantes: que la respuesta del sistema sea la esperada o que no lo sea.
EC 2	Escenario 2: Terminación del caso de uso	1- Se tienen el conjunto de datos que serán objeto de la graficación. 2-Se decide no continuar con la graficación	No se mostrará la gráfica de los datos.	Se especifica que sucedió al realizar la prueba, pueden existir dos variantes: que la respuesta del sistema sea la esperada o que no lo sea.

Tabla 10 Caso de prueba CU Graficar datos en forma de barra

Matriz de Casos de Prueba. Caso de Uso Graficar Datos en forma de línea.

Id del	Escenario	Datos	Respuesta del	Resultado de la prueba

escenario			**Sistema**	
EC 1	Escenario1: Graficación exitosa de datos	Se tienen el conjunto de datos que serán objeto de la graficación.	Se mostrará la gráfica de línea correspondiente a los datos.	Se especifica que sucedió al realizar la prueba, pueden existir dos variantes: que la respuesta del sistema sea la esperada o que no lo sea.
EC 2	Escenario 2: Terminación del caso de uso	1- Se tienen el conjunto de datos que serán objeto de la graficación. 2-Se decide no continuar con la graficación	No se mostrará la gráfica de los datos.	Se especifica que sucedió al realizar la prueba, pueden existir dos variantes: que la respuesta del sistema sea la esperada o que no lo sea.

Tabla 11 Caso de prueba CU Graficar datos en forma de línea.

Id del escenario	**Escenario**	**Datos**	**Parámetro**	**Respuesta del Sistema**	**Resultado de la prueba**
EC 1	Escenario 1: Graficación exitosa de datos	Se tienen el conjunto de datos que serán objeto de la graficación.	Contiene el parámetro de la gráfica de pastel que será graficada.	Se mostrará la gráfica de pastel correspondiente a los datos.	Se especifica que sucedió al realizar la prueba, pueden existir dos variantes: que la respuesta del sistema sea la esperada o que no lo sea.
EC 2	Escenario 2: Error en la entrada de parámetro	Se tienen el conjunto de datos que serán objeto de la graficación.	El campo que recoge el nombre del parámetro está vacio.	El sistema emite un mensaje para que se escoja un parámetro.	Se especifica que sucedió al realizar la prueba, pueden existir dos variantes: que la respuesta del sistema sea la esperada o que no lo sea.

Tabla 12 Caso de prueba CU Graficar datos en forma de pastel

Con los siguientes casos de prueba se procede a la realización de las pruebas, algunas de las cuales se muestran en los anexos, con valores reales según las especificaciones de cada campo en la matriz de casos de prueba. El objetivo principal de esta etapa es la detección de errores, los cuales serán registrados para su posterior corrección.

4.9 Conclusiones.

Los autores consideran que lo expuesto en el presente capítulo resume los principales elementos para la realización del sistema propuesto. Todo lo referente a la concepción del sistema y los principales elementos de programación junto con las pruebas realizadas constituyen elementos de gran importancia para futuras versiones del graficador.

CAPÍTULO 5. Estudio de factibilidad.

5.1Introducción.

El estudio de la factibilidad de un proyecto es una de las tareas que implica mayor importancia, con el estudio de la misma se determina si será recomendable la realización del mismo, apoyándose en los elementos de costo y beneficios del mismo.

5.2 Planificación.

Para la planificación del mismo se utilizará la técnica de estimación de esfuerzo y tiempo de desarrollo por Puntos de Casos de Uso. En este método intervienen los actores y los casos de uso a los cuales se les asigna una complejidad, además también participan factores técnicos y de entorno. A continuación se desarrolla el método mencionado para la estimación del proyecto.

5.2.1 Calculando los puntos de casos de uso sin ajustar.

UUCP: Puntos de Casos de Uso sin ajustar.

UAW: Factor de Peso de los Actores sin ajustar.

UUCW: Factor de Peso de los Casos de Uso sin ajustar.

UUCP=UAW+UUCW

La complejidad de los actores se establece teniendo en cuenta en primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema.

Tipo de Actor	Descripción	Factor de Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación.	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2
Complejo	Una persona que interactúa con el sistema a través de una interfaz gráfica.	3

Tabla 13 Peso de los actores

En el sistema se tiene la siguiente información:

Nombre del actor	Complejidad
Usuario	Persona que interactúa con el sistema a través de una interfaz gráfica. Complejidad 3.

Tabla 14 Peso de los actores del sistema.

UAW=3*1=3.

Para calcular el factor de peso de los casos de uso (UUCW):

1. Si el tipo de caso de uso es simple contiene de 1 a 3 transacciones luego su factor de peso es 5.
2. Si el tipo de caso de uso es medio contiene de 4 a 7 transacciones luego su factor de peso es 10.
3. Si el tipo de caso de uso es complejo contiene más de 8 transacciones luego su factor de peso es 15.

Casos de uso	Peso
Gestionar Datos	10
Graficar datos en forma de barra	5
Graficar datos en forma de línea	5
Graficar datos en forma de pastel	5
Construir ejes coordenados	5
Salvar gráfica	5

Tabla 15 Peso de los casos de uso del sistema

Siguiendo la siguiente tabla se tiene: UUCW=1*10+5*5=35.

Entonces UUCP= UUCW+ UAW =35+3=38.

5.2.2 Calcular los puntos de casos de uso ajustados (UCP).

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar éste valor mediante la siguiente ecuación:

UCP = UUCP * TCF * EF

UCP = Puntos de Casos de Usos Ajustados

TCF = Factor de Complejidad Técnica

EF = Factor de Ambiente

Para Calcular TCF

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante y se sigue la siguiente ecuación.

TCF = 0.6 + 0.01 * Σ (Peso * Valor$_i$)

Factor	Descripción	Peso	Valor
T1	Sistema distribuido.	2	0
T2	Objetivos de performance o tiempo de respuesta.	1	4
T3	Eficiencia del usuario final.	1	3
T4	Procesamiento interno complejo	1	3
T5	El código debe ser reutilizable	1	5
T6	Facilidad de instalación	0,5	4
T7	Facilidad de uso	0,5	3
T8	Portabilidad	2	5
T9	Facilidad de cambio	1	4
T10	Concurrencia	1	4
T11	Incluye objetivos especiales de seguridad	1	0
T12	Provee acceso directo a terceras partes	1	4
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	1

Tabla 16 Factor de complejidad técnica.

TCF=0.6+0.01*41.5=1.015

Para calcular EF se sigue el mismo procedimiento anterior, en este caso en la siguiente tabla

EF = 1.4 - 0.03 * Σ (Peso$_i$ * Valor$_i$) (Donde Valor es un número del 0 al 5)

Factor	Descripción	Peso	Valor
E1	Familiaridad con el modelo de proyecto utilizado	1,5	2
E2	Experiencia en la aplicación	0,5	0
E3	Experiencia en orientación a objetos	1	5
E4	Capacidad del analista líder	0,5	4
E5	Motivación	1	5
E6	Estabilidad de los requerimientos	2	2
E7	Personal a tiempo compartido	-1	0
E8	Dificultad del lenguaje de programación	-1	3

Tabla 17 Factor de ambiente.

EF = 1.4 - 0.03*16=0.92.

Con los Puntos de Casos de Uso sin ajustar (**UUCP**) y los Factores de Complejidad Técnica (**TCF**) y de Ambiente (**EF**) calculados, entonces:

UCP = UUCP * TCF * EF

UCP = 33*1.015*0.92= 35.4844

Calculando el esfuerzo de la fase de implementación.

Donde:

E: Esfuerzo estimado en horas-hombre.

UCP: Puntos de Casos de Uso ajustados.

CF: Factor de conversión.

Convertir los Puntos de Casos de Uso Ajustados a Esfuerzo de desarrollo.

Se contabilizan cuántos factores de los que afectan al factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.

Se contabilizan cuántos factores de los que afectan al factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.

Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.

Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre.

Tenemos que los factores que afectan al factor ambiente son 3 por tanto el CF=28 horas-hombre/Punto de casos de Uso.

E=UCP*CF=30.8154*28= 993,5632 horas –hombre

Este esfuerzo representa un porciento del esfuerzo total según la tabla de estimación para los diferentes flujos de trabajo del proyecto. El esfuerzo total conociendo que un mes equivale a 192 horas de trabajo.

Actividad	Porcentaje %	Horas-Hombres
Análisis	5	124,1954
Diseño	25	620,977
Implementación	40	993,5632
Pruebas	20	496,7816
Sobrecarga (otras actividades)	10	248,3908
Total	100	2483,908

Tabla 18 Esfuerzo del proyecto

ET= 2483,908 horas –hombre= 12,9370208 mese-hombre. Esto significa que una persona realizaría el trabajo completo en 13 meses aproximadamente.

5.3 Costos

Teniendo en cuenta que serán dos personas trabajando en el proyecto, la duración del mismo sería 6 meses y dos semanas, lo cual para un salario medio de 100 pesos mensuales por cada persona, se obtiene el costo estimado del proyecto:

CE=12.93*100= 1293.7020

5.4 Beneficios tangibles e intangibles.

La realización del sistema conlleva varios beneficios, donde lo principal será poseer un graficador propio de datos en dos dimensiones para su utilización fundamentalmente en los softwares de simulación del polo del mismo nombre de la facultad 9, sin embargo se podrá utilizar indistintamente en cualquier aplicación que necesite graficación. Además se conocerá la arquitectura del sistema posibilitando así cualquier cambio en las funcionalidades de este como la ampliación de las mismas en un futuro.

5.5 Conclusiones.

Se considera totalmente factible llevar a la práctica el desarrollo del sistema pues el costo es pequeño y los beneficios que se obtendrán serán considerables. No se necesitará depender de otros sistemas de esta misma naturaleza, los cuales en varias ocasiones deben ser usados bajo licencias, lo cual dificulta el trabajo que se necesite realizar.

CONCLUSIONES

El desarrollo de la investigación se basó fundamentalmente en la buena planificación de las actividades, así como del estudio profundo realizado en el área de la graficación y del uso de la tecnología Java para la implementación del graficador, con la culminación de la misma se cumplieron los objetivos trazados al comienzo de su desarrollo:

Se obtuvo el graficador de datos en el plano para su utilización en los distintos softwares de simulación.

Se conoce la arquitectura del sistema, producto de aplicar la metodología de desarrollo RUP, la cual soporta el desarrollo del sistema.

La arquitectura obtenida sirve para la construcción de otros graficadores de datos en el plano en otras plataformas de trabajo.

RECOMENDACIONES

1. Realizar pruebas de caja blanca para probar más exhaustivamente el producto obtenido.
2. Ampliar las funcionalidades del sistema, basándose en otros tipos de gráfica que existen, las cuales responden a otros detalles específicos según sus características.
3. Incluir la solución propuesta en los simuladores que se desarrollan en la facultad para valorar su eficiencia y efectividad, así como su impacto en la visualización de los datos arrojados en los procesos de simulación.
4. Utilizar la solución en cualquier otro software en el cual se necesite de la graficación de datos en el plano para la mejor comprensión de los mismos.

Referencia Bibliográfica.

1. D.M., H. *Análisis Y Simulación De Procesos.* Reverte, 2007, ISBN 8429172351 84-291-7235-1.

2. *G D S S I M* Lima-Perú: de 2008]. Disponible en: http://gdssim.com/.

3. PARRA, R. A. M. *Tipos de Simulación* de 2008]. Disponible en: http://docencia.50webs.com/simula02.htm.

4. NIEBEL B., F. A. *Ingeniería Industrial, Métodos Estándares y Diseño del Trabajo.* 3 ed. México: Alfaomega, 2001.

5. DÜRSTELER, J. C. *Texto, tablas y gráficos.* 2002, Disponible en: http://www.infovis.net/printMag.php?num=108&lang=1.

6. ---. *Presentaciones Gráficas.* 2002, Disponible en: http://www.infovis.net/printMag.php?num=73&lang=1.

7. ---. *Gráficos de Barras.* 2004, Disponible en: http://www.infovis.net/printFicha.php?rec=revista&num=157&lang=1&palabra=Graficos%20de%20barra.

8. *Gráficos de líneas* de 2008]. Disponible en: http://technet.microsoft.com/es-es/library/ms159640.aspx.

9. GUTIÉRREZ, L. R. A. *Apuntes sobre Representación Gráfica.* 2003, Disponible en: http://www.cecam.sld.cu/pages/rcim/revista_4/articulos_html/rene.htm.

10. *Gráficos de burbujas* Disponible en: http://technet.microsoft.com/es-es/library/ms155868.aspx.

11. *Tipos de gráficos disponibles.* Disponible en: http://office.microsoft.com/es-es/help/HA012337373082.aspx.

12. *.netCHARTING Enterprise Edition* Disponible en: http://www.duamu.com/re/script/999/id/10824/scripts--netcharting-enterprise-edition.html.

13. *PocketGear* sitio de descarga. Disponible en: http://classic.pocketgear.com/software_detail.asp?id=9556.

14. *Información importante sobre Swiftchart: chart and graph java applet* de 2008]. Disponible en: http://textver.filehungry.com/spanish/product/windows_software/programming/specialized_tools/swiftchart__chart_and_graph_java_applet.

15. MATTL. *La Definición de Software Libre* Equipo de traductores a español de GNU. Disponible en: http://www.gnu.org/philosophy/free-sw.es.html.

16. JAMES RUNBAUGH, I. J., GRADY BOOCH. *El Lenguaje Unificado de Modelado. Manual de Referencia.* Addison Wesley, 2000, Disponible en: http://bibliodoc.uci.cu/pdf/reg03050.pdf.

17. IVAR JACOBSON, G. B., JAMES RUMBAUGH. El Proceso Unificado de Desarrollo de Software. 2000, nº Disponible en: http://bibliodoc.uci.cu/pdf/reg00060.pdf.

18. *DiaRio de un LiNuX3Ro* de 2008]. Disponible en: http://albertjh.cymaho.com/?p=196.

19. MYATT, A. *Pro NetBeans™ IDE 6 Rich Client Platform.* apress, 2008. ISBN 978-1-59059-895-5.

20. FLANAGAN, D. *JAVA En Pocas Palabras.* 1998, Disponible en: http://bibliodoc.uci.cu/pdf/reg01329.pdf. ISBN 1998. 970-10-2070-7.

Bibliografía

1. Beck, Kent. 1999. *Extreme programming explained: embrace change.* Boston : Addison-Wesley Longman Publishing Co, 1999.

2. Benjamín NIEBEL, Andris FREIVALDS. 2001. *Métodos, Estándares y Diseño del Trabajo.* México : Alfaomega, 2001. 970-15-0993-5.

3. Douglas Bell, Doug Bell, Mike Parr, Alfonso Vidal Romero Elizondo, Guillermo Levine Gutiérrez. 2003. *Java Para Estudiantes.* s.l. : Pearson Educación, 2003. 9702601444.

4. Dürsteler, Juan C. 2002. *Presentaciones Gráficas.* 73, 2002.

5. Ekcel, Bruce. 1998. *Piensa en Java.* s.l. : PRENTICE HALL, 1998.g

6. Flanagan, David. 1998. *JAVA En Pocas Palabras.* Mexico : s.n., 1998. 970-10-2070-7.

7. Harvey M. Deitel, Paul J. Deitel, Alfonso Vidal Romero. 2004. *Cómo programar en Java.* s.l. : Pearson Educación, 2004. 9702605180.

8. HERRAMIENTAS CASE. [En línea] http://ceds.nauta.es/informes/case04.htm.

9. Ivar Jacobson, Grady Booch,James Rumbaugh. 1999. *El Proceso Unificado de Desarrollo de Software.* 1999.

10. James Rumbaugh, Ivar Jacobson,Graddy Booch. 2000. *El Lenguaje Unificado de Modelado.Manual de Referencia.* 2000.

11. Khawar Zaman Ahmed, Cary E. Umrysh. 2001. *Developing Enterprise Java Applications with J2EE and UML.* s.l. : Addison-Wesley Pub Co, 2001.

12. Larman, Craig. 1997. *UML y Patrones.* 1997.

13. Myatt, Adam. 2008. *Pro NetBeans™ IDE 6 Rich Client Platform.* 2008.

14. Peralta, Mario. *ESTIMACIÓN DEL ESFUERZO BASADA EN CASOS DE USO.*

15. RAFAE TORRES ROBLES L, J. JAVIER CASTRO ARELLANO. 2003**.** *ANALISIS Y SIMULACION DE PROCESOS DE REFINACION DEL PETROLEO.* s.l. : Alfaomega, 2003. 970150836X.

16. Schmuller, Joseph. 2000. *APRENDIENDO UML EN 24 HORAS.* México : PEARSON EDUCACION, 2000. 968-444- 463-X.

17. S. Pressman, Roger. *Ingeniería de Software. Un Enfoque Práctico.* Madrid: s.n., 2001. (Beck, 1999)

18. Thomas H. Naylor, R. Bustamante**.** Simulación. [En línea] [Consultado el: 10 de enero de 2008.] http://es.wikipedia.org/wiki/Simulaci%C3%B3n.

19. Zukowski, John. 2003**.** *Programación Java 2 J2SE 1.4.* Madrid : Ediciones Anaya Multimedia, 2003. 84-415-1559-X.

20. 2004. Adictos Al Trabajo. [En línea] 02 de 02 de 2004. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=vp.

21. 2008. monografías.com. *Simulación de procesos.* [En línea] 2008. http://www.monografias.com/trabajos6/sipro/sipro.shtml.

Anexos.

Anexo I. Interfaces

Interfaz Principal

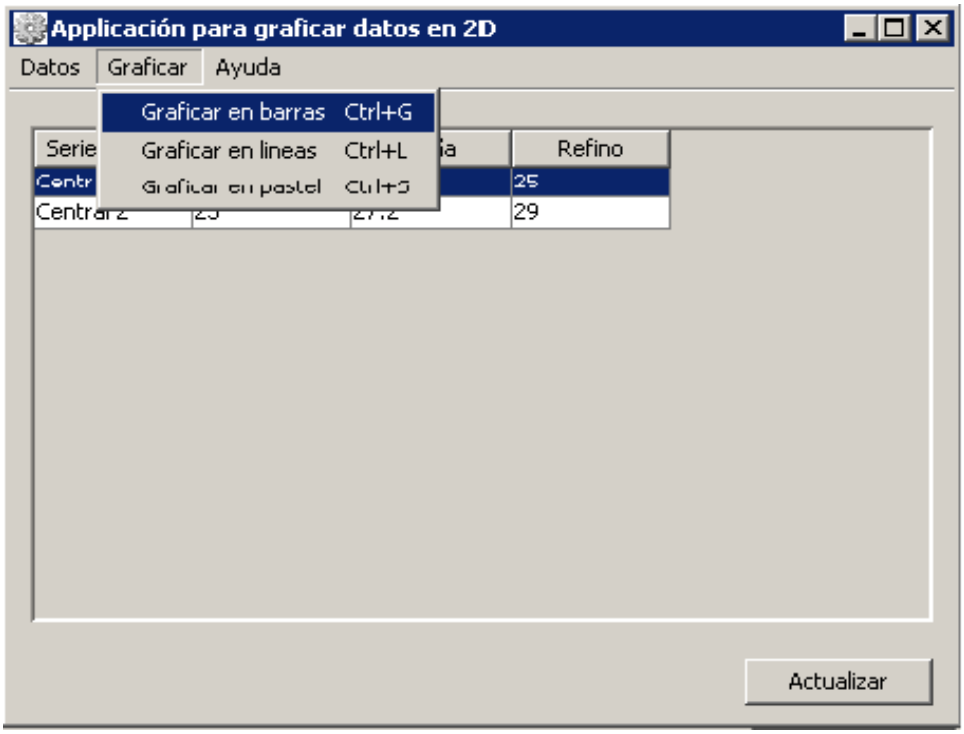


Figura 18 Interfaz Principal

Interfaz Títulos

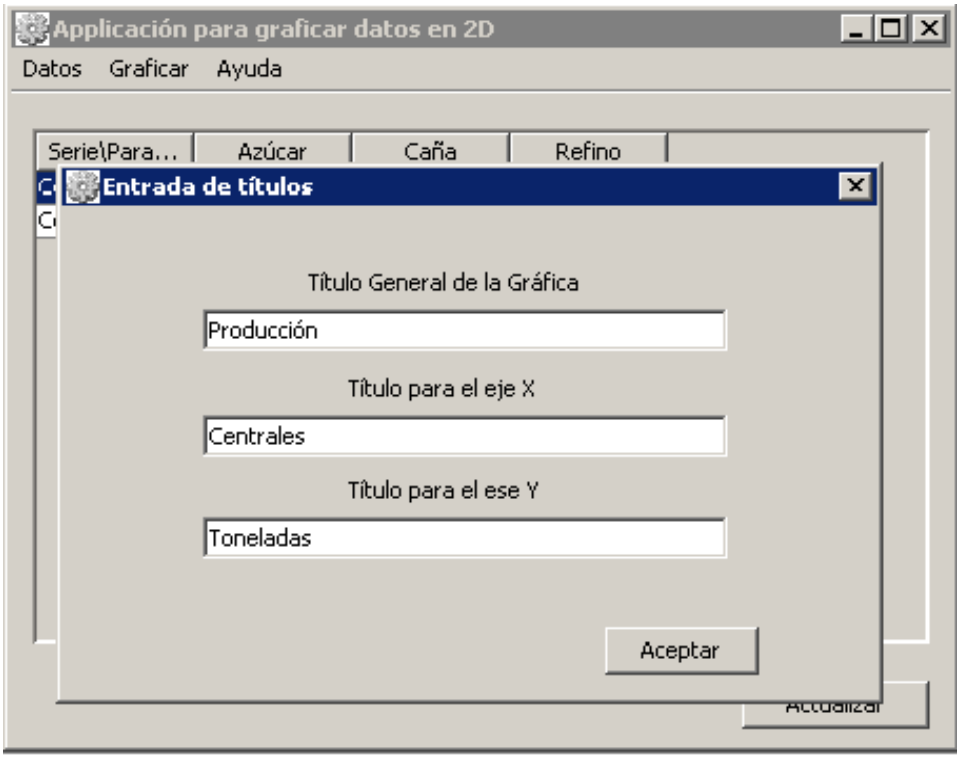


Figura 19 Interfaz para títulos

Interfaz para la graficación

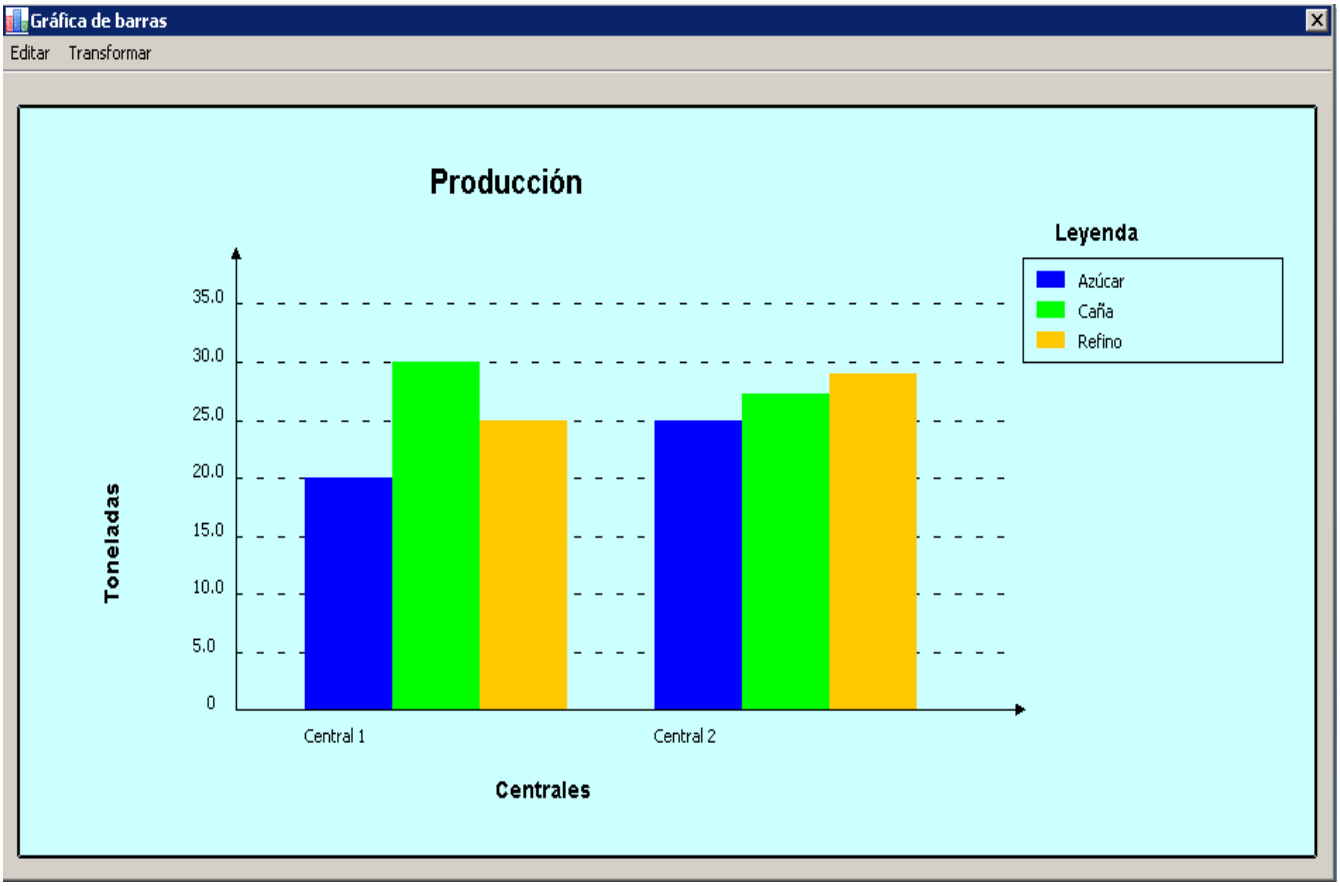


Figura 20 Gráfica de barra.

Interfaz para la graficación

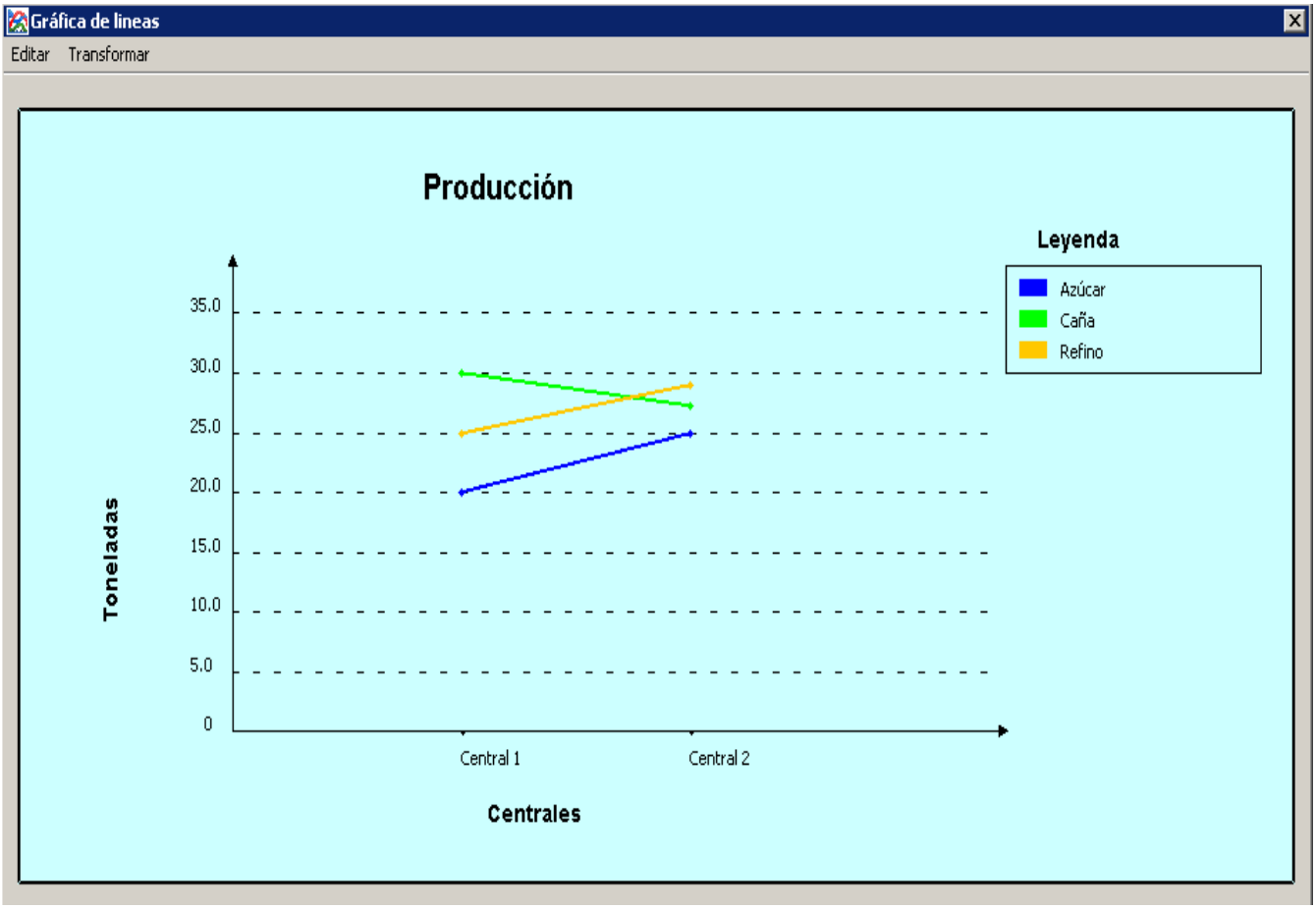


Figura 21 Gráfica de líneas

Interfaz para la graficación

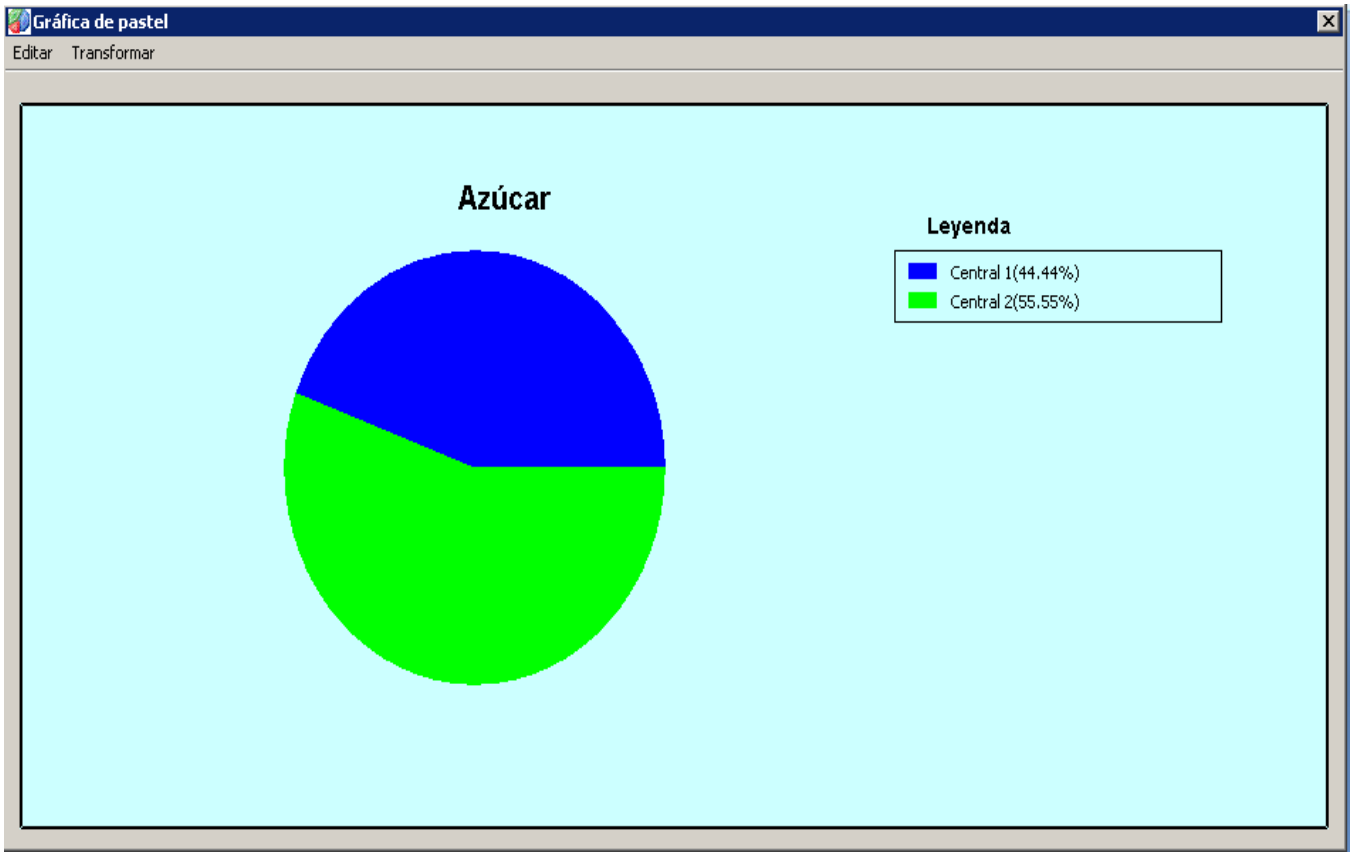


Figura 22 Gráfica de pastel

Anexo 2.Pruebas.

Caso de Uso Manejar Datos.

Id del escenario	Escenario	Parámetros	Series	Datos	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario1: Manejo exitoso de datos	"Azúcar", "Caña", "Refino"	"Central 1" "Central 2"	20,30,25 25,27.2,29	El sistema tendrá los datos necesarios para la graficación.	El sistema tiene los datos necesarios para la graficación.

Tabla 19 Caso de prueba del CU Manejar datos (EC 1)

Id del escenario	Escenario	Parámetros	Series	Datos	Respuesta del Sistema	Resultado de la prueba
EC 2	Escenario 2: Ausencia de datos.	"Azúcar", "Caña", "Refino"	"Central 1" "Central 2"	20,30,25	El sistema emite un mensaje de anomalía.	Se muestra un mensaje para que introduzca los datos que faltan.

Tabla 20 Caso de prueba del CU Manejar datos (EC 2)

Id del escenario	Escenario	Parámetros	Series	Datos	Respuesta del Sistema	Resultado de la prueba
EC 2	Escenario 2: Ausencia de datos.	"Azúcar"	" "	Aún no se han introducido	El sistema emite un mensaje de anomalía.	El resultado es un mensaje para que se llene el

						campo vacio.

Tabla 21 Caso de prueba del CU Manejar datos (EC 2)

Id del escenario	Escenario	Parámetros	Series	Datos	Respuesta del Sistema	Resultado de la prueba
EC 2	Escenario 2: Ausencia de datos.	" "	"Central1"	Aún no se han introducido	El sistema emite un mensaje de anomalía.	El resultado es un mensaje para que se llene el campo vacio.

Tabla 22 Caso de prueba del CU Manejar datos (EC 2)

Caso de Uso Graficar Datos en forma de barra.

Id del escenario	Escenario	Datos	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario1: Graficación exitosa de datos	20,30,25 25,27.2,29	Se mostrará la gráfica de barra correspondiente a los datos.	Se obtuvo la gráfica de barra esperada según los datos.

Tabla 23 Caso de prueba del CU Graficar datos en forma de barra (EC 1)

Id del escenario	Escenario	Datos	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario1: Graficación exitosa de datos	23.2,-25,-25 -12,-21.2,29	Se mostrará la gráfica de barra correspondiente a los datos.	Se obtuvo la gráfica de barra esperada según los datos.

Tabla 24 Caso de prueba del CU Graficar datos en forma de barra (EC 1)

Caso de Uso Graficar Datos en forma de línea.

Id del escenario	Escenario	Datos	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario1: Graficación exitosa de datos	20,30,25 25,27.2,29	Se mostrará la gráfica de barra correspondiente a los datos.	Se obtuvo la gráfica de línea esperada según los datos.

Tabla 25 Caso de prueba del CU Graficar datos en forma de línea (EC 1)

Id del escenario	Escenario	Datos	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario1: Graficación exitosa de datos	23.2,-25,-25 -12,-21.2,29	Se mostrará la gráfica de barra correspondiente a los datos.	Se obtuvo la gráfica de línea esperada según los datos.

Tabla 26 Caso de prueba del CU Graficar datos en forma de línea (EC 1)

Caso de Uso Graficar Datos en forma de pastel.

Id del escenario	Escenario	Datos	Parámetro	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario1: Graficación exitosa de datos	20,30,25 25,27.2,29	"Azúcar"	Se mostrará la gráfica de pastel correspondiente a los datos.	Se mostró la gráfica de pastel correspondiente al parámetro.

Tabla 27 Caso de prueba del CU Graficar datos en forma de pastel (EC 1)

Id del escenario	Escenario	Datos	Parámetro	Respuesta del Sistema	Resultado de la prueba
EC 1	Escenario1: Graficación exitosa de datos	10, 12.3 -2.3,15 -5.6,-36	“Temperatura”	Se mostrará la gráfica de pastel correspondiente a los datos.	Se mostró la gráfica de pastel correspondiente al parámetro.

Tabla 28 Caso de prueba del CU Graficar datos en forma de pastel (EC 1)

Id del escenario	Escenario	Datos	Parámetro	Respuesta del Sistema	Resultado de la prueba
EC 2	Escenario 2: Error en la entrada parámetro	20,30,25 25,27.2,29	“ ”	El sistema emite un mensaje para que escoja un parámetro.	Se muestra un mensaje para seleccionar un parámetro.

Tabla 29 Caso de prueba del CU Graficar datos en forma de barra (EC 2)

Adobe Systems: es una empresa de software que destaca por sus programas de edición de páginas web, vídeo e imagen digital.

Drag & Drop: operación de pinchar y arrastrar que facilita la utilización del ratón para realizar operaciones de forma más sencilla y rápida.

Modo landscape: es una forma horizontal de configurar la pantalla para la mejor observación de los elementos.

Pocket Excel: es la versión móvil de Microsoft de Excel.

Plugin(s): es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

Release: se refiere a un producto final, preparado para lanzarse como versión definitiva.