Universidad Del Biobío
Facultad De Ciencias Empresariales
Ingeniería Ejecución en Computación e Informática

"PROYECT GRAFO MAX"

Integrantes:
Víctor Jarpa Hermosilla
Héctor Poblete Concha
José Bastías Hernández
Andrés Garcés Alarcón

Profesora: Brunny Troncoso

Asignatura: Paradigmas de Programación

Fecha de entrega: 09-01-2013

1-INTRODUCCIÓN

El Presente Informe da a conocer la documentación sobre el proyecto semestral de desarrollo de software “Proyet Grafo Max” para la asignatura Paradigmas de Programación.

Proyet Grafo Max posee una variedad de funciones que tiene la funcionalidad de crear, editar y manipular grafos, a través de una interfaz para que el usuario interactúe fácilmente con el software y pueda manipular grafos de manera eficiente y sencilla.

Este informe define primeramente la “especificación de requerimientos de software”. En esta parteencontraremos los “Alcances” donde se describe las características del software que lo diferencian de otros, “Objetivos del software” se describe los objetivos que debe cumplir el software en forma general y específica, “Descripción global del producto” se analiza la “interfaz de usuario” y la “interfaz de software” y finalmente los “Requerimientos específicos” donde se presenta los Requerimientos Funcionales del sistema.

Luego se profundizará el “Análisis” del software, en la cual se presentan distintos diagramas (Diagrama Casos de Usos, Diagrama de Secuencias) y un modelamiento de datos la cual se utiliza un diagrama de clases.

A continuación el informe explicara el “Diseño” del software. Se analiza el “Diseño interfaz y Navegación”.

En la sección de “Pruebas”, contiene las “Especificaciones de las pruebas” que indica ejemplos que se utilizó para probar el software y verificar errores, y “Conclusiones de Prueba” que indica los resultados obtenidos.

Siguiendo el informe se presenta los “algoritmos”, donde se indica en pseudolenguaje los algoritmos implementados, para desarrollar y solucionar diversos problemas.

Finalmente vienen las secciones de “Conclusión” donde se presentan conclusiones de los pasos, herramientas y lenguajes utilizados en el proyecto, “Líneas Futuras de desarrollo” se presentan posibles expansiones, mejoras que se podrán realizar, una visión a futuro del producto y finalizando la “Bibliografía” la cual da a conocer referencias y bibliografía.

2- ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

2.1- Alcances

Proyect Grafo Max es un software desarrollado para que los usuarios puedan crear, editar, manipular y guardar grafos en forma sencilla y segura.

Para iniciar la manipulación de este software se debe crear un nodo la cual al instante el usuario podrá disponer de la matriz de adyacencia, información y algoritmos sobre los caminos que posee cada nodo.

2.2 Objetivo del software

El objetivo principal del software es lacreación de un editor simple de grafos confuncionalidades básicas para crear, modificar y el guardar los trabajos realizados en este editor. El software posee opciones gráficas para el dibujo de grafos como tambiénpara la edición de los nodos y de sus partes, esto se refiere al cambio de forma de los nodos, etiquetas, posiciones, peso de las aristas, colores y tamaños. El software genera de manera automática la matriz de adyacencia asociada al grafo creado. Finalmente entrega información sobre el grafo construido la cual da a conoceral usuario el grado de sus vértices, si el grafo es conexo, dirigido, bipartito y los caminos: mínimo, eulerianos y hamiltonianos.

2.3- Descripción Global del Producto

2.3.1- Interfaz de usuario

Proyect Grafo Max posee una interfaz fácil de utilizar pensado para cualquier tipo de usuario la cual contiene una gran variedad de herramientas para crear y modificar grafos,además de iconos para mostrar resultados de los grafos creados y todo esto ubicados en el sector izquierdo de la pantalla(barra menú icono), la cual además consta de información de la utilidad de cada icono para guiar a los usuarios sobre todo a los más inexpertos o desconocedores de este software.

En el sector derecho de la pantalla se muestran los distintos resultados de los grafos creados como matriz de adyacencia, caminos (mínimos, eulerianos y hamiltonianos).

El software cuenta con un menú desplegable la cual presenta todas las funciones que posee el software, nombrando algunas de ellas por ejemplos sería crear, guardar mostrar matriz, editar, herramientas(obtener diámetro, colorear, entre otras), algoritmos y finalmente ayuda.

2.4- Requerimientos Específicos.

2.4.1- Requerimientos Funcionales Del Sistema

Id	Nombre	Descripción
R01	Editar Grafos	
R. 01.1	Abrir Grafo	El software abre archivo existente
R. 01.2	Nuevo grafo	El software crea un nuevo archivo
R. 01.3	Guardar Grafo	El software guarda un archivo
R. 01.4	Eliminar Grafo	El software elimina un archivo
R. 02	Editar Vértices	Edita los vértices del grafo
R. 02.1	Agregar Vértice	Agrega un nuevo vértice al grafo
R. 02.2	Eliminar Vértice	Elimina un vértice del grafo
R. 02.3	Desplazar vértice	Desplaza un vértice en distintas posiciones
R. 02.4	Modificar Vértice	Modifica color,tamaño y forma del vértice
R. 03	Editar Arista	Edita las aristas del grafo
R. 03.1	Agregar Arista	Agrega una nueva artistaal grafo
R. 03.2	Eliminar Arista	Elimina unaarista del grafo
R. 03.3	Desplazar Arista	Desplaza una arista en distintas posiciones
R. 03.4	Modificar Arista	Modifica color,tamaño y forma delaarista
R. 04	Imprimir	Imprime el grafo creado
R. 05	Visualizar/ Ocultar	Habilita o deshabilita: 1-matriz de adyacencia 2-cuadriculado de fondo para lienzo del dibujo 3-tabla de grados asociado a cada vértice 4-las propiedades que el grafo cumple
R.06	Gestionar Acciones	Deshace o rehace acciones
R.07	Guardar Como	Exporta un grafo a diversos formatos
R.08	Cortar, Copiar y Pegar	Corta, copia y/o pega secciones seleccionados del grafo
R.09	Mostrar Bipartito	Muestra gráficamente y además en una tabla los vértices que componen cada conjunto y señala si es grafo bipartito o bipartito completo
R10	Mostrar Ciclos Eulerianos	Muestra todos los ciclos Eulerianos que existen en el grafo, indicando distancia de recorrido
R.11	Mostrar caminos Hamiltonianos	Muestra todos los ciclos Hamiltonianos que existen en el grafo, indicando distancia de recorrido
R.12	Mostrar Caminos/ciclos	Presenta gráficamente (paso a paso, pintando las aristas) los caminos o ciclos que existen entre dos vértices
R13	Edición de grafos	Permite ver propiedades del grafo: tabla de grados, completitud, bipartito, conexo, ciclos eulerianos, caminos Hamiltonianos, alinear grafo.
R.14	Mostrar si es conexo	Muestra si el grafo es conexo y si lo es de que tipo de conexo se trata(fuertemente, unilateralmente, débilmente).
R.15	Colorear grafos	Presenta gráficamente el coloreo de un grafo
R.16	Informar Diametro de Grafo	Muestra la mayor distancia entro todos los pares de nodos

R17	Mostrar Árbol Binario	Informa si el grafo se trata de un árbol binario o no, si lo es lo alinea y muestra altura y recorridos tales como: PreOrden, InOrden, PostOrden.
R.18	Aplicar algoritmo Dijkstra	Determina el camino mas corto en un grafo.
R.19	Aplicar Algoritmo Kruskal	Encuentra un árbol recubridor mínimo en un grafo conexo ponderado.
R.20	Aplicar algoritmo Prim	Encuentra un árbol recubridor mínimo en un grafo conexo ponderado.

3- ANALISIS

3.1 Diagramas de casos de usos

3.1.1 Actores

El único actor que se identifica en el software es el usuario, el cual tiene como rol el de utilizar el programa adecuadamente. Sus conocimientos requeridos serán los básicos en la teoría de grafos, para así poder utilizar de mejor forma las funciones del software. El usuario sólo tiene acceso a las funcionalidades implementadas desde la interfaz.

3.1.2-Casos de usos y descripción

Diagrama de Caso de uso: Insertar Vértice.

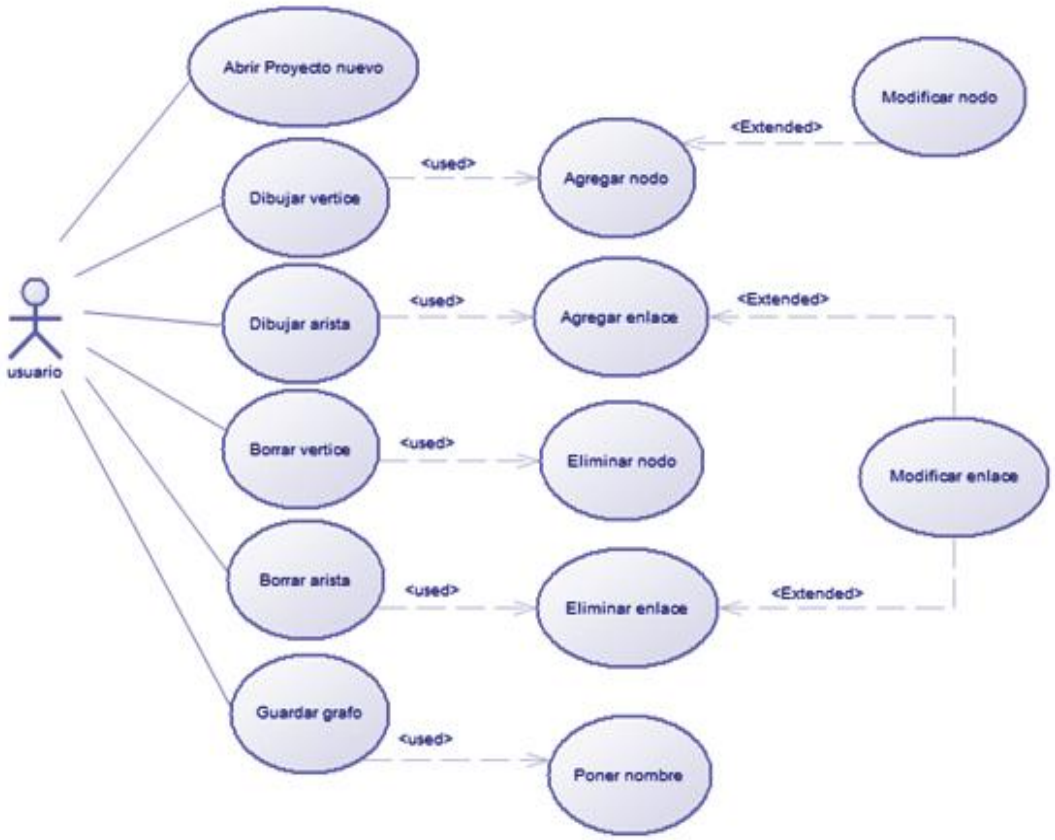


Diagrama de Caso de uso: Insertar Vértice.

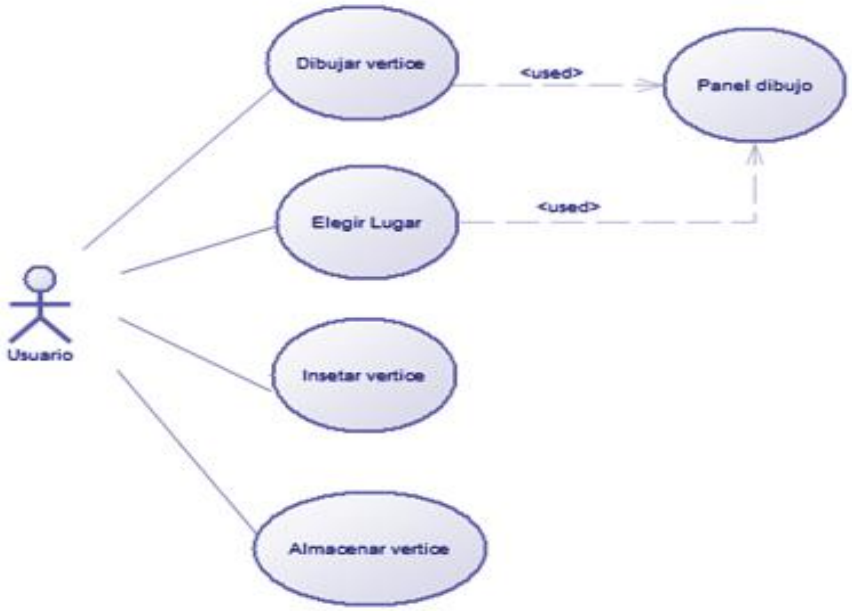
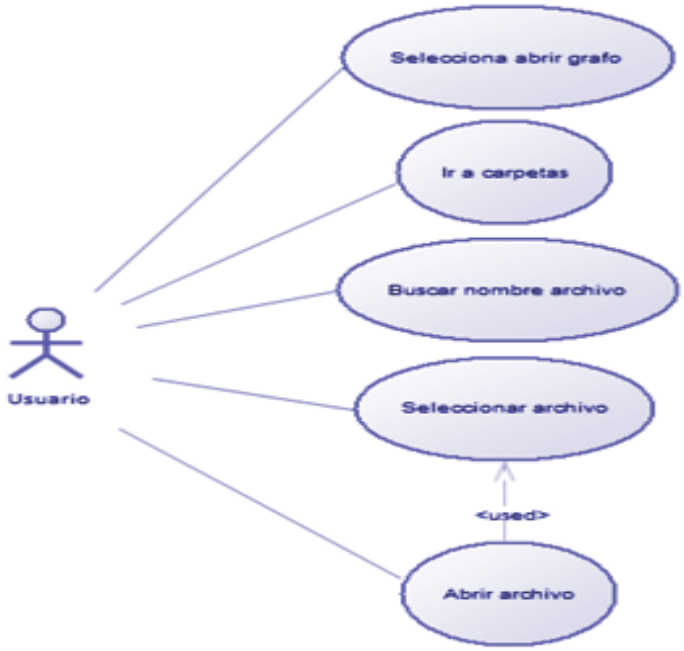


Diagrama de Caso de uso: Abrir Grafo.



3.1.3-Especificación casos de usos

Caso de Uso: <Dibujar Vértice>

- Descripción: Dibuja en el lienzo un ovalo que representa un nodo.
- Pre-Condiciones: En el lienzo deben existir menos de 10 nodos.
- Flujo de Eventos Básicos: El usuario hace clic con el mouse en el botón insertar nodo y luego hace clic en el lienzo y donde hace clic aparece un nodo.
- Flujo de Eventos Alternativo: Si existen 10 nodos en el lienzo, aunque el usuario haga clic en el botón crear nodo, y luego clic en el lienzo no ocurrirá nada.
- Post-Condiciones: Al agregar un nodo se inserta de manera automática a una matriz de enteros.

Caso de Uso: <Dibujar Arista>

- Descripción: Dibuja en el lienzo una arista que conectan nodos
- Pre-Condiciones: Como mínimo deben existir 2 nodos para que se pueda dibujar una arista.
- Flujo de Eventos Básicos: El usuario hace clic con el ratón en el botón insertar arista y luego presiona el botón izquierdo sobre el nodo de entrada y arrastra hasta el nodo de salida donde suelta el botón izquierdo del ratón.
- Flujo de Eventos Alternativo: Si existe entre dos nodos una arista, aunque el usuario haga clic en el botón crear arista, y luego realice todo el procedimiento para insertar una nueva, el programa no realiza nada.
- Post-Condiciones: Al agregar una arista se modifica automáticamente la matriz de enteros.

Caso de Uso: <Insertar Vértice>

- Descripción: Inserta un vértice en el panel para poder ensamblar las aristas y unir los demás nodos hasta formar un grafo.
- Pre-Condiciones: En el lienzo debe existir menos de 10 vértices
- Flujo de Eventos Básicos: El usuario hace clic con el mouse en el botón insertar vértice y luego hace clic en el lienzo y donde hace este clic aparece un vértice.
- Flujo de Eventos Alternativo: Si posee ya 10 vértices creados y quiere hacer más, cuando el usuario haga clic en el botón crear vértice, no ocurrirá nada.

- Post-Condiciones: El sistema debe almacenar vértices en el grafo.

Caso de Uso: <Almacenar Vértice>

- Descripción: Almacena un vértice.
- Pre-Condiciones: En el lienzo deben existir menos de 10 vértices para poder almacenar.
- Flujo de Eventos Básicos: El usuario hace clic con el mouse en el botón insertar vértice y luego aparece en el lienzo, después para que no se pierda presiona en guardar y queda almacenado el vértice y no se pierde.
- Flujo de Eventos Alternativo: Si existen 10 vértices en el lienzo, aunque el usuario haga clic en el botón crear nodo, y luego clic en el lienzo no ocurrirá nada y solo se podrán almacenar esos 10.
- Post-Condiciones: El sistema debe almacenar el vértice en el grafo.

Caso de Uso: <Buscar Archivo>

- Descripción: Busca un archivo que contenga un grafo en especifico.
- Pre-Condiciones: Se debe buscar por el nombre específico que tenga el archivo.
- Flujo de Eventos Básicos: El usuario hace clic con el mouse donde dice abrir archivo y realiza una búsqueda por carpetas donde se encuentre el archivo que se desee abrir.
- Flujo de Eventos Alternativo: Si el usuario no busca el archivo indicado nunca lo va ha encontrar.
- Post-Condiciones: El sistema debe reconocer el archivo indicado por el usuario.

Caso de Uso: <Abrir Archivo>

- Descripción: Abre un archivo seleccionado por el usuario.
- Pre-Condiciones: El sistema muestra una ventana que permite al usuario elegir el archivo a abrir.
- Flujo de Eventos Básicos: El usuario selecciona el botón de la barra de herramientas o desde el menú de archivo y luego abrir.
- Flujo de Eventos Alternativo: El sistema abre el archivo y representa el contenido del archivo en el panel de dibujo.

- Post-Condiciones: El usuario selecciona el archivo que desea abrir.

3.2Diagramas de Secuencias

Diagrama de Secuencia: Dibujar Grafo.

Diagrama de Secuencia: Insertar Vértice.

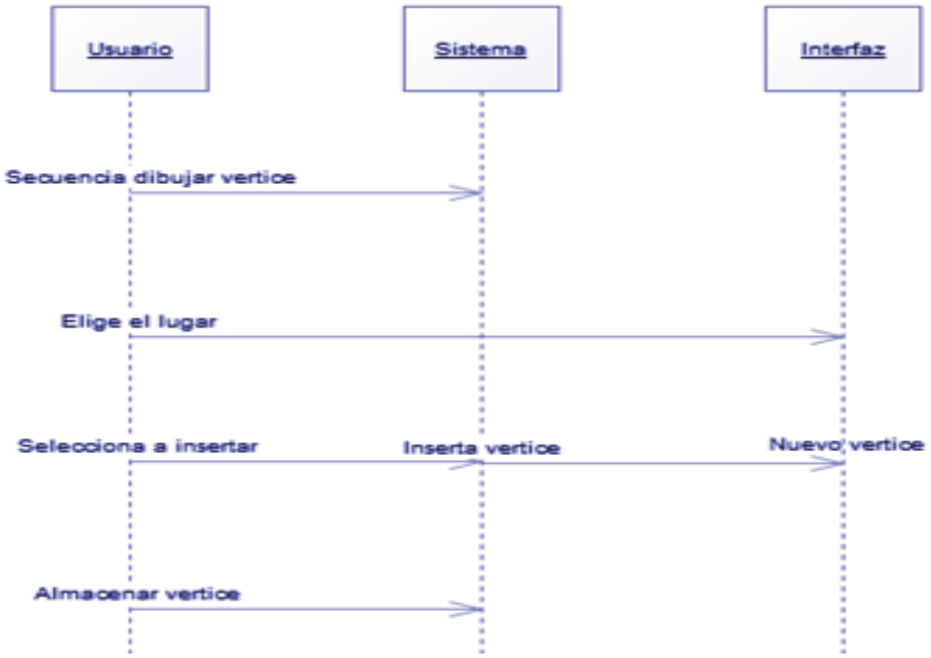
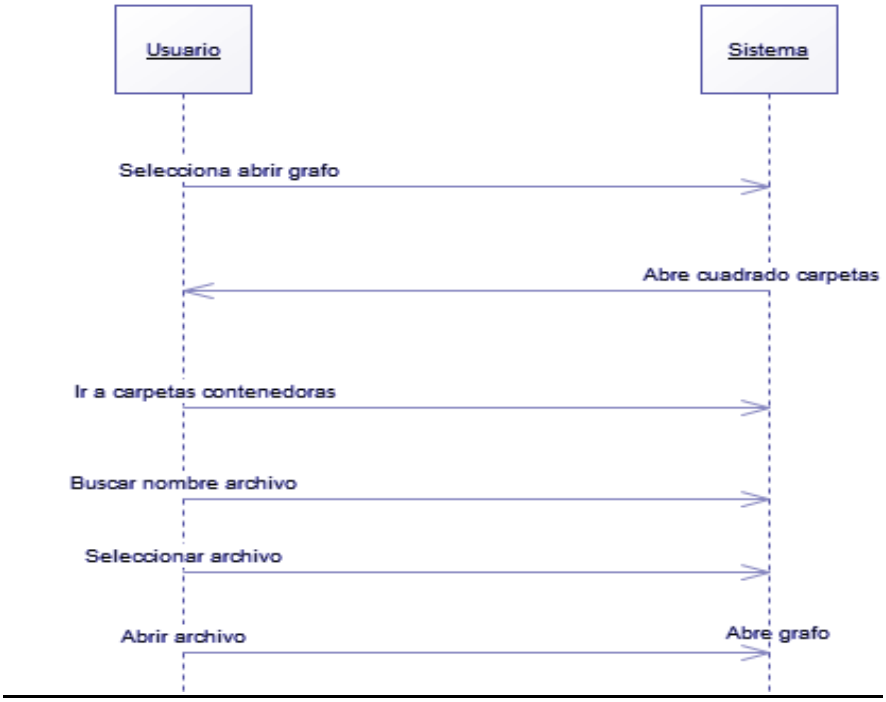


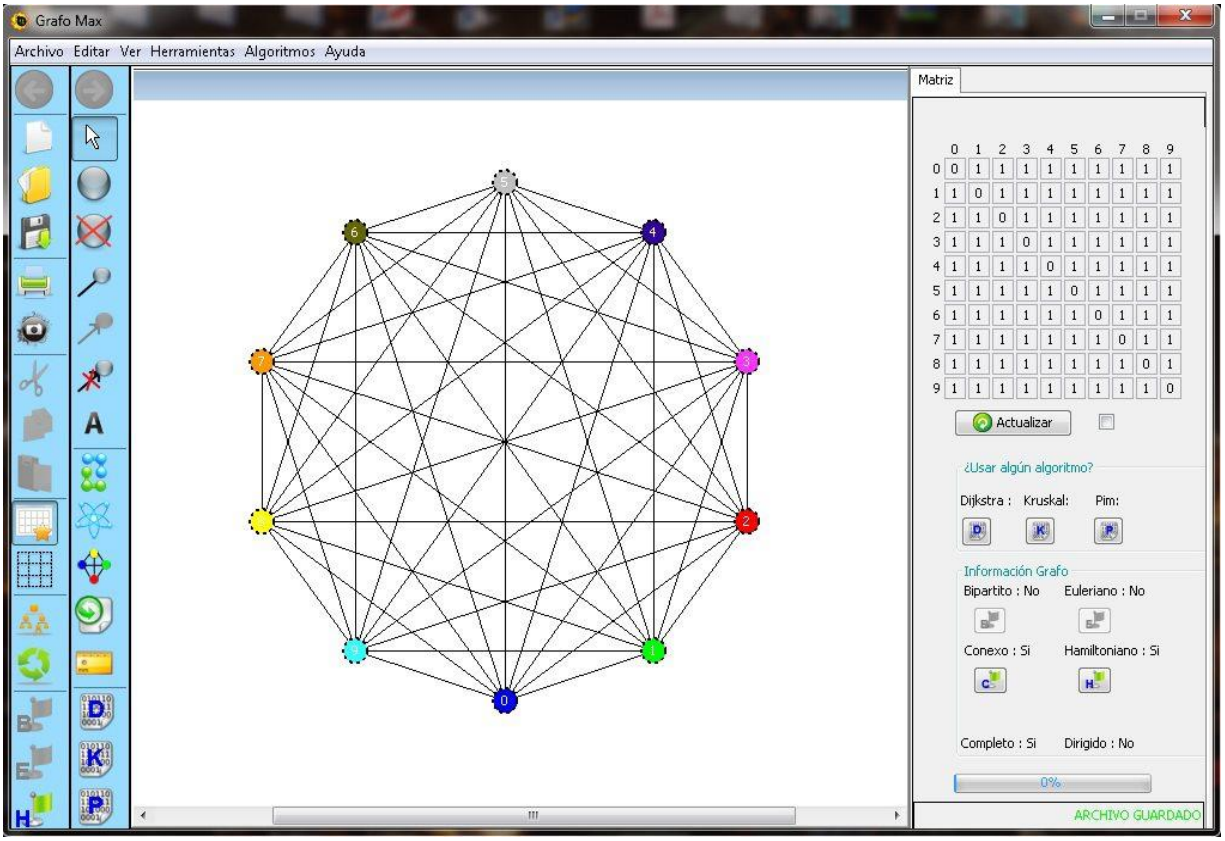
Diagrama de Secuencia: Abrir Grafo.



4- DISEÑO

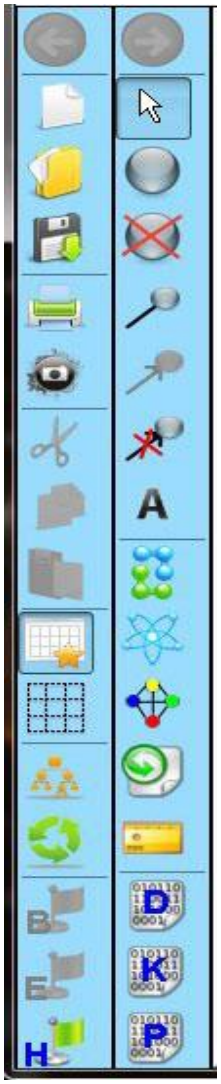
4.1- Diseño interfaz y navegación

La siguiente imagen muestra la interfaz del software.



El usuario puede crear diversos nodos (10 máximo), de distintos colores y en distintas posición como se muestra en la imagen, al igual que puede crear aristas

Barra Menú Icono



En esta barra se presentan todas las herramientas que se utilizan para crear adecuadamente un grafo. Como se aprecia los iconos denotan claramente su función, estos sirve de gran ayuda a todos los usuarios que utilicen este software.

En este sector de la interfaz se muestran diversos resultados de un grafo. El usuario dispondrá de la matriz de adyacencia, podrá utilizar tres distintos algoritmos los cuales son Dijkstra, Kruskal y Prim.

Finalmente el software presenta la información de un grafo, la cual muestra si es Bipartito, Euleriano, Conexo y Dirigido.

5-PRUEBAS

5.1- Especificación de pruebas

Características a probar	Objetivo de la prueba	Técnicas para la definición de casos de prueba	Actividades de prueba	Criterios de cumplimiento
Aristas	Comprobara la precisión al seleccionar una arista	Presentar en distintos ángulos las aristas	Crear 4 nodos y 3 aristas para luego que con el mouse el usuario pueda ponerlas donde quiera sin problema	Para que esto de resultado deberá seleccionarse solo cuando el mouse se posiciones sobre la arista
Cambiar Color del nodo	Corroborar que al cambiar color funcione correctamente	Realizar la prueba de la función en distintos nodos	Crear nodos para poder cambiar su color, y realizar todas las funciones correspondientes para poder realizar la acción	Para dar por finalizada la prueba deberá mantener el color cambiado para cada situación.
Cambiar nombre al nodo	Corroborar el completo funcionamiento al cambiar una etiqueta	Etiquetar y cambiar etiquetas a varios nodos del grafo	Poder crear nodos y poder cambiar su etiqueta sin problema	Que salga correctamente el nombre designado por el usuario
Guardar en formato distinto	Corroborar que se guarde correctamente en el formato seleccionado	Guardar distintos grafos hechos en distintos formatos	Crear varios grafos y almacenarlos en una formato de imagen sea en jpg, gif, etc	Para dar esta prueba por superada se debe comprobar la existencia del archivo de imagen creada.

5.2- Conclusiones de Prueba

La conclusión al realizar las pruebas seleccionadas, es que algunas pruebas no daban resultados satisfactorios ya sea con la entrega de resultados erróneos o simplemente el software se caía, por lo tanto modifican los algoritmos hasta obtener los resultados deseados. Finalmente los usuarios podrán ver un software eficiente y no sufrirá de problemas como los inconvenientes que ocurrieron durante la creación de este.

6-ALGORITMOS

6.1-Algoritmo de Dijkstra(pseudocódigo).

```
void dijkstra(int fuente, Grafo g, matriz costoAristas){
   S = {fuente}
   n = cantidadVertices(g)
   for(i=0; i//vector de costos
      costos[i] = costoAristas[fuente, i] //inicializo el costo desde la fuente hasta los otros
vértices (        infinito en el caso que no exista relación) //vector de camino final (por si se
necesita reconstruir el camino más corto)
      caminoRecorrido[i] = fuente
      visitados[i] = 0; //inicializa un vector de nodos visitados
    }
    visitados[fuente] = 1;//marca a la fuente como visitada
   for(i=0; i
      if(i != fuente){
         w = obtenerMinimoVerticeDisponible(g) //esto es : el minimo del conjunto (V - S)
         S = S U {w} //agrega el vértice utilizado
         visitados[w] = 1 //lo marca como visitado
        for(j=0; j
            if(visitados[j] != 1){
                //que es menos costoso, ir directamente -> costos [j] o ir mediante un w
intermedio
              if(costos[w] + costoAristas[w, j] < costos[j]){
                    //es menos costoso ir mediante w
                costos[j] = costos[w] + costoAristas[w, j];
                caminoRecorrido[j] = w //agrego el nodo utilizado
                }
             }
          }
      }
  }
```

6.2 Algoritmo de Kruskal

```
Kruskal (G)
E(1)=0, E(2)=  todos los Arcos del grafo G
    Mientras E(1) contenga menos de n-1 arcos y E(2)=0 do
      De los arcos de E(2) seleccionar el de menor coste -->e(ij)
      E(2)= E(2) - {e(ij)}
         Si V(i), V(j) no están en el mismo árbol entonces
           juntar los árboles de  V(i) y de V(j) en uno sólo
         end Si
    end do
Fin del algoritmo
```

6.3 Algoritmo de Prim

```
A={s};
     D[s]=0;
     D[v]=cost(s,v) para todo v en V-A; // infinito si el arco no existe
     while(A!=V)
     {
      Encontrar v en V-A tal que D[v] es mínimo;
      Agregar v a A;
      for(todo w tal que (v,w) está en E)
        D[w]=min(D[w],D[v]+cost(v,w));
    }
```

7-CONCLUSIONES

Java muestra una nueva forma de diseñar proyectos y resolver problemas, a través de la orientación a objetos. Tiene una interacción mas atractiva y más cómoda tanto para al programados como para el usuario.

En este proyectos se logra adquirir mayor conocimientos sobre grafos, su comportamiento y funcionalidad y la experiencia de desarrollar un software mas avanzado en comparación a los ya creados a lo largo de los estudios. El desafío de implementar sobre un proyecto ya creado por otros compañeros, permite aprender más sobre las maneras de realizar cambios dentro de un contexto incompleto y aprovechar las herramientas eficientes que disponemos ya sea códigos de compañeros anteriores o simplemente pseudocódigos extraídos de internet.

A lo largo de este informe se analiza el software ya sea algunos pseudocódigos, interfaz, y algo muy importante como las pruebas que se realizaron al software paratener como resultado un software de calidad.

8-LINEAS FUTURAS DE DESARROLLO

Hoy en día la programación en JAVA es muy importante por sobre otros lenguajes por su modularidad y por su enfoque de programación orientado a objetos.

Si se piensa en una visión a futuro de este software sería la implementación de nuevas funcionalidades que serán aprendidas durante el transcurso de la formación profesional, para hacer de este proyecto un software mas avanzado e innovador.

9- BIBLIOGRAFIA

Deitel & Deitel, *Como programar en Java*, 7ª edición, 2009

Aprendiendo UML en 24 horas, Joseph Schmuller.

http://www.wikipedia.org/