****

**Complejidad Algorítmica**

**Entregable Final**

**Profesor:** Luis Martin Canaval Sánchez

**Sección:**

**Integrantes:**

* Eduardo Arias
* Daniel Carnero
* Rafael Samanez

2018-02

**Índice**

**1. Introducción** 3

**2. Marco conceptual** 3

**3. Requisitos** 5

**4. Diagrama de clases de entidades principales** 6

**5. Ejecución** 7

**6. Bibliografía** 13

1. **Introducción**
   1. **Problema**

El proyecto del curso busca implementar Implementa funciones y (o) clases para generar el grafo de centros poblados de Perú a partir del dataset proporcionado por el ministerio de educación, mediante los algoritmos aprendidos en el curso, en el caso de nuestro proyecto, por ejemplo:

* MST (Prim)
* Bellman-Ford
* Warshall

Dada una colección de ciudades y el costo del viaje entre cada par de ellas, el problema del vendedor ambulante, o TSP, es encontrar la forma más económica de visitar todas las ciudades y regresar a su punto de partida. En la versión estándar que estudiamos, los costos de viaje son simétricos en el sentido de que viajar de la ciudad X a la ciudad Y cuesta tanto como viajar de Y a X.

La simplicidad de la declaración del problema es engañosa: el TSP es uno de los problemas más estudiados en matemáticas computacionales y, sin embargo, no se conoce ningún método de solución eficaz para el caso general. De hecho, la resolución del TSP resolvería el problema P versus NP y obtendría un premio de $ 1,000,000 del Clay Mathematics Institute.

**Objetivos**

* Desarrollar el proyecto con estrategias apropiadas para resolver el problema a nivel algorítmico, y descomponer el problema en subtareas.
* Establecer un cronograma de trabajo que permita distribuir y desarrollar de manera más eficiente el trabajo en equipo
* Hacer uso de buenas prácticas de programación para tener una mejor optimización y evidenciar el rendimiento en la ejecución del algoritmo y la visualización de los mismos gráficamente, experimentando con los datasets y que evidencien una buena optimización al utilizarse.
* Definir un producto final para poder ser evaluado en el curso

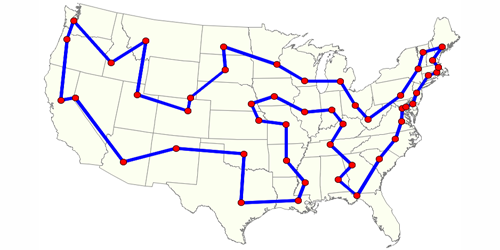
1. **Marco conceptual**

* **Traveling Salesman Problem**

Aunque aún se desconoce la complejidad del TSP, durante más de 50 años, su estudio ha llevado a mejorar los métodos de solución en muchas áreas de la optimización matemática y programación.

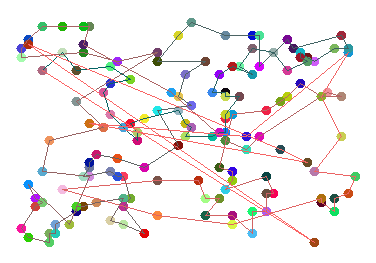
Los problemas matemáticos relacionados con el problema del vendedor ambulante fueron tratados en la década de 1800 por el matemático irlandés Sir William Rowan Hamilton y por el matemático británico Thomas Penyngton Kirkman. La imagen de abajo es una fotografía del juego Icosian de Hamilton que requiere que los jugadores completen recorridos a través de los 20 puntos utilizando solo las conexiones especificadas. Una buena discusión de los primeros trabajos de Hamilton y Kirkman se puede encontrar en el libro Graph Theory 1736-1936 por N. L. Biggs, E. K. LLoyd y R. J. Wilson, Clarendon Press, Oxford, 1976.

La forma general del TSP parece haber sido estudiada por primera vez por matemáticos a partir de la década de 1930 por Karl Menger en Viena y Harvard. El problema fue luego promovido por Hassler Whitney y Merrill Flood en Princeton. Se puede encontrar un tratamiento detallado de la conexión entre Menger y Whitney, y el crecimiento del TSP como tema de estudio en el artículo de Alexander Schrijver "En la historia de la optimización combinatoria (hasta 1960)".

. 

Gran parte del trabajo sobre el TSP está motivado por su uso como plataforma para el estudio de métodos generales que se pueden aplicar a una amplia gama de problemas de optimización discreta. Sin embargo, esto no quiere decir que el TSP no encuentre aplicaciones en muchos campos. De hecho, las numerosas aplicaciones directas del TSP dan vida al área de investigación y ayudan a dirigir el trabajo futuro.

El TSP surge naturalmente como un subproblema en muchas aplicaciones de transporte y logística, por ejemplo, el problema de organizar las rutas de los autobuses escolares para recoger a los niños en un distrito escolar. Esta aplicación de bus tiene una importancia histórica importante para el TSP, ya que motivó a Merrill Flood, uno de los pioneros de la investigación de TSP en la década de 1940. Una segunda aplicación de TSP de la década de 1940 involucró el transporte de equipo agrícola de un lugar a otro para probar el suelo, lo que llevó a estudios matemáticos en Bengala por P. C. Mahalanobis y en Iowa por R. J. Jessen. Las aplicaciones más recientes incluyen la programación de llamadas de servicio en las empresas de cable, la entrega de comidas a personas confinadas en el hogar, la programación de grúas apiladoras en los almacenes, el enrutamiento de camiones para la recogida de paquetes y otros.



Aunque las aplicaciones de transporte son el entorno más natural para el TSP, la simplicidad del modelo ha llevado a muchas aplicaciones interesantes en otras áreas. Un ejemplo clásico es la programación de una máquina para taladrar agujeros en una placa de circuito u otro objeto. En este caso, los orificios a perforar son las ciudades, y el costo del viaje es el tiempo que toma mover el cabezal de perforación de un orificio a otro. La tecnología para la perforación varía de una industria a otra, pero siempre que el tiempo de viaje del dispositivo de perforación sea una parte importante del proceso general de fabricación, el TSP puede desempeñar un papel en la reducción de costos.

Para dar al lector una muestra de algunas aplicaciones actuales del TSP, proporcionamos en las siguientes páginas una lista de algunos de los trabajos aplicados (y no tan aplicados, pero aún así divertidos) que involucran módulos de la biblioteca TSP de Concorde.

1. **Requisitos**

Para la definición de requisitos utilizamos el criterio de especificación de ***Historias de Usuario***:

**Historia:** Definir y diseñar la estructura del programa

**Como:** Estudiante

**Quiero:** Definir las principales clases y estructuras de datos que utilizare para implementar la utilización de lo dataSet, la interfaz gráfica.

**Porque:** Con esto poder obtener la información necesaria para así aplicar después los algoritmos de búsqueda en grafos.

**Criterios de Aceptación:**

* Utilizar los data set proporcionados
* Lenguaje de programación adecuado

**Historia:** Implementar una interfaz de usuario

**Como:** Estudiante

**Quiero:** Visualizar una interfaz que me permita abrir la dataSet, ver el contenido de éste, y realizar pruebas.

**Porque:** Ello permitirá que el usuario pueda tener una experiencia interactiva con el aplicativo

**Criterios de Aceptación:**

* Mostrar información relevante
* Interfaz no muy sobrecargada
* Utilizar la data set completa

**Historia:** Implementación de Grafos

**Como:** Estudiante

**Quiero:** Visualizar en la pantalla los grafos los cuales permitirán la visualización e input para

**Porque:** Con los grafos demostraré la utilización de los algoritmos que se implementaran para la solución del problema de TSP

**Criterios de Aceptación:**

* Datos de ubicación del grafo con nombre de ciudad o centro poblado
* Desplazamiento en el mapa para la visualización correcta de grafos

**Historia:** Obtención de resultados e información luego de procesar el algoritmo

**Como:** Estudiante

**Quiero:** Visualizar una opción que me permita ver o obtener la información resultante al utilizar el algoritmo implementado.

**Porque:** Ello evidenciara el uso de los algoritmos aprendidos en el curso.

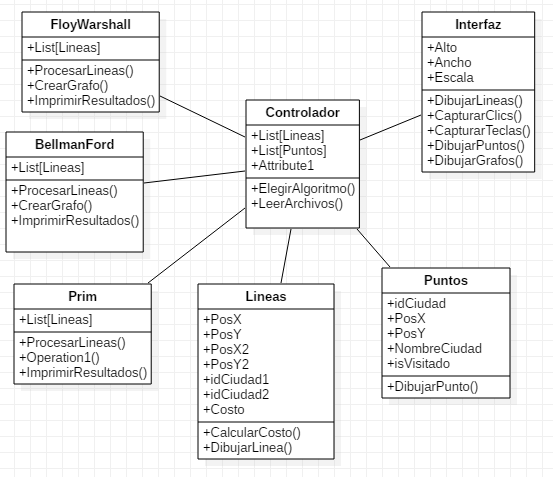
**Criterios de Aceptación:**

* Se muestra la información en pantalla
* Costos, ciudad y otros criterios

**Requisitos no funcionales:**

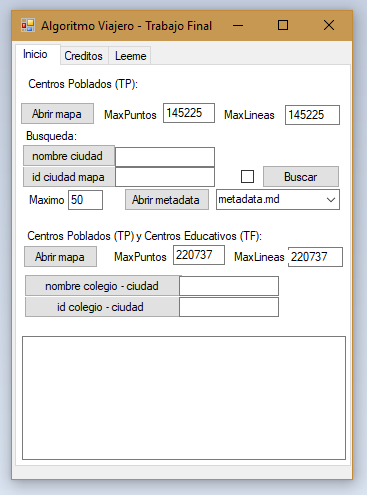
* Se usará Windows Form Aplication para el diseño de interfaz del aplicativo
* Se usará un programa en consola para el procesamientos de dataSets .
* Se hará uso de la programación orientada a objetos en el Lenguaje C++ y C#.

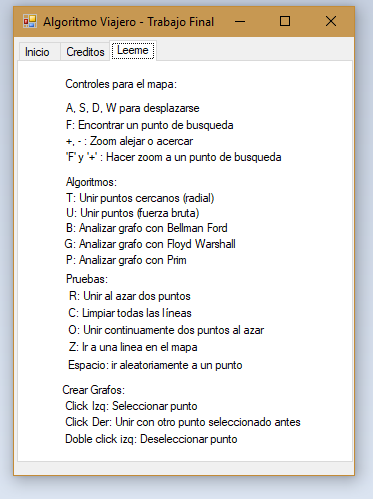
1. **Diagrama de clases de entidades principales**



1. **Ejecución** 
   1. **Diseño de interfaz de usuario**

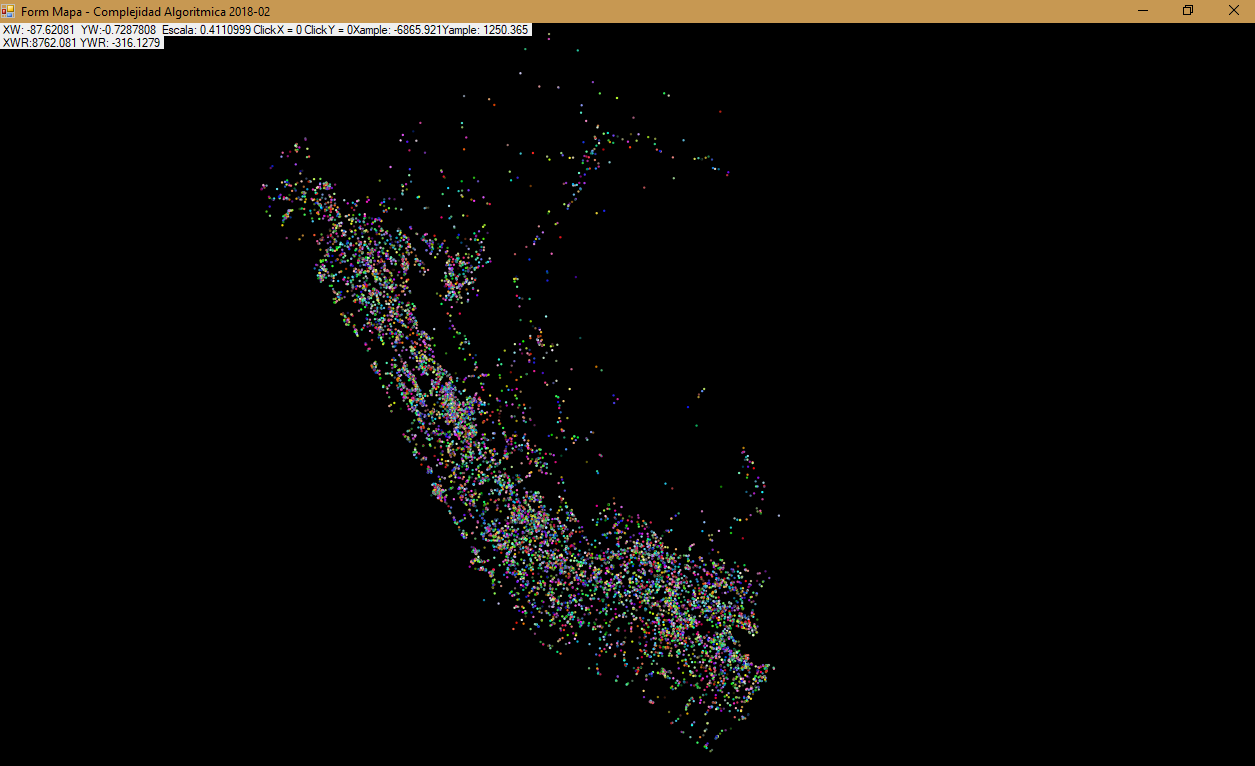
**Menu principal:**

** **

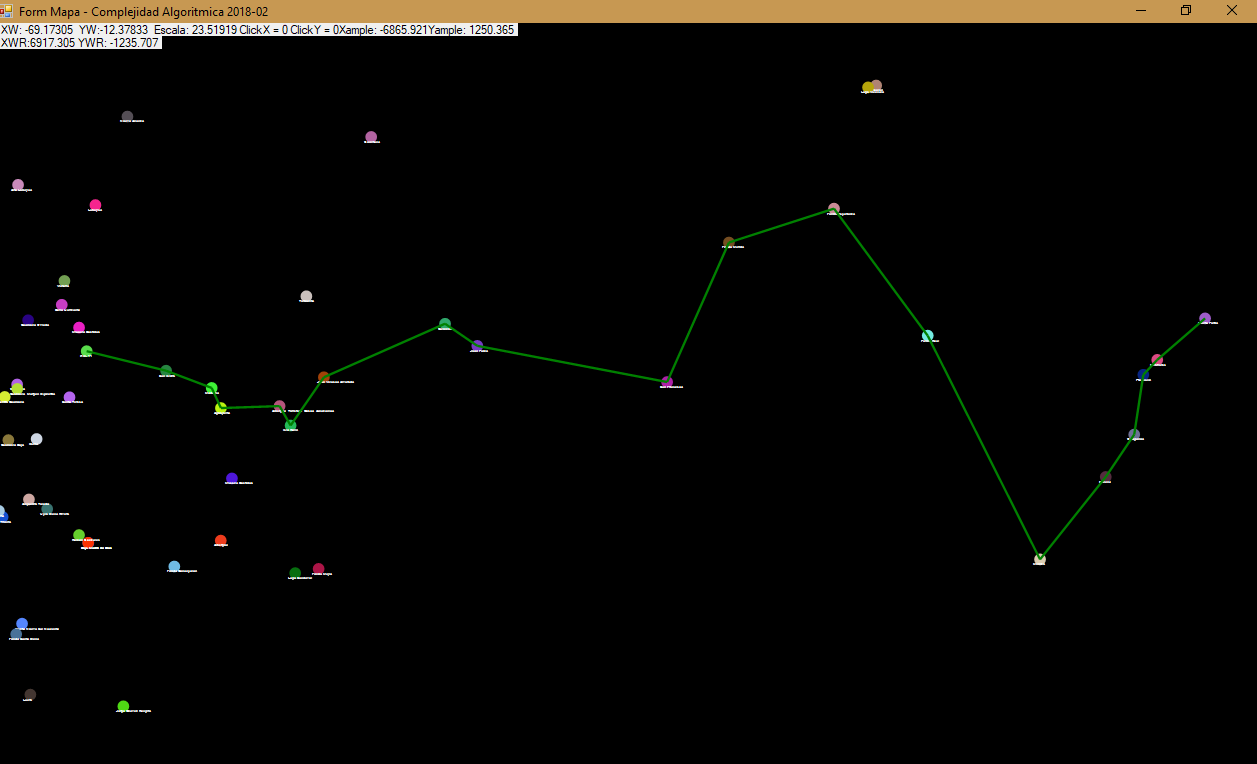
****

**Interfaz gráfica:**

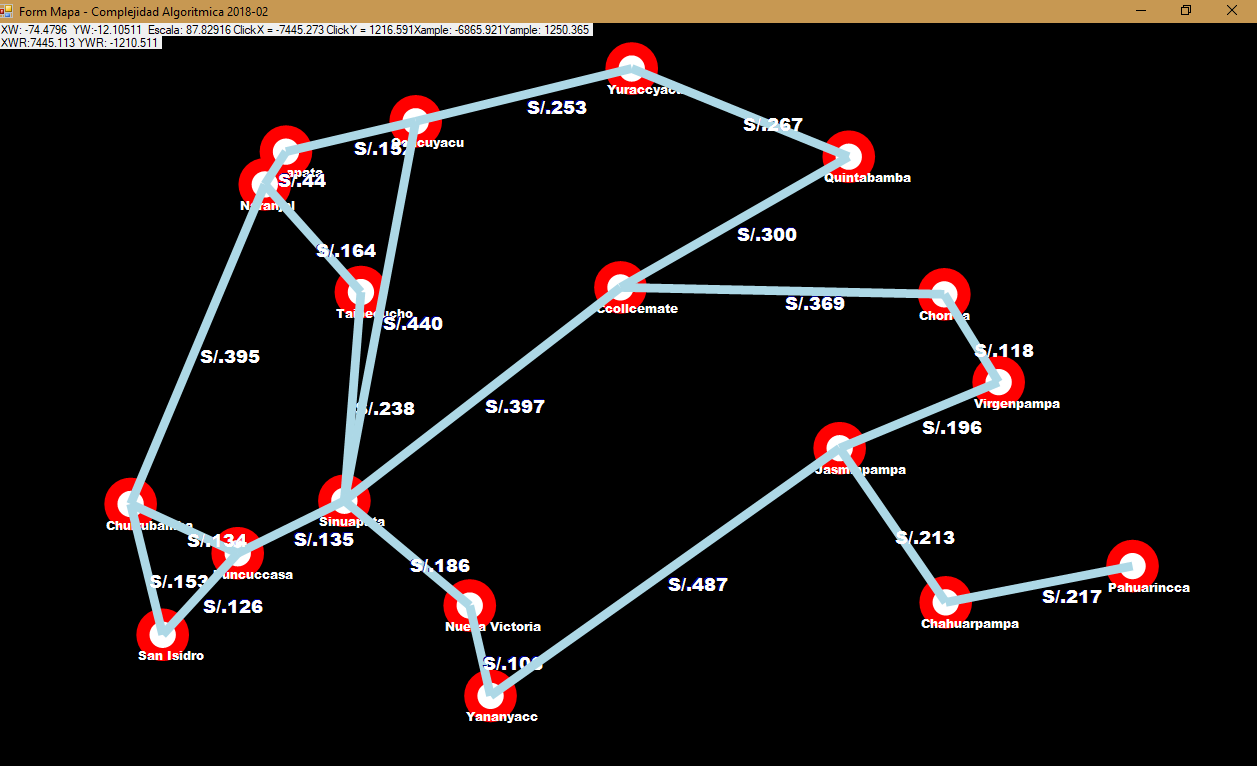
**Data set individual:**

****

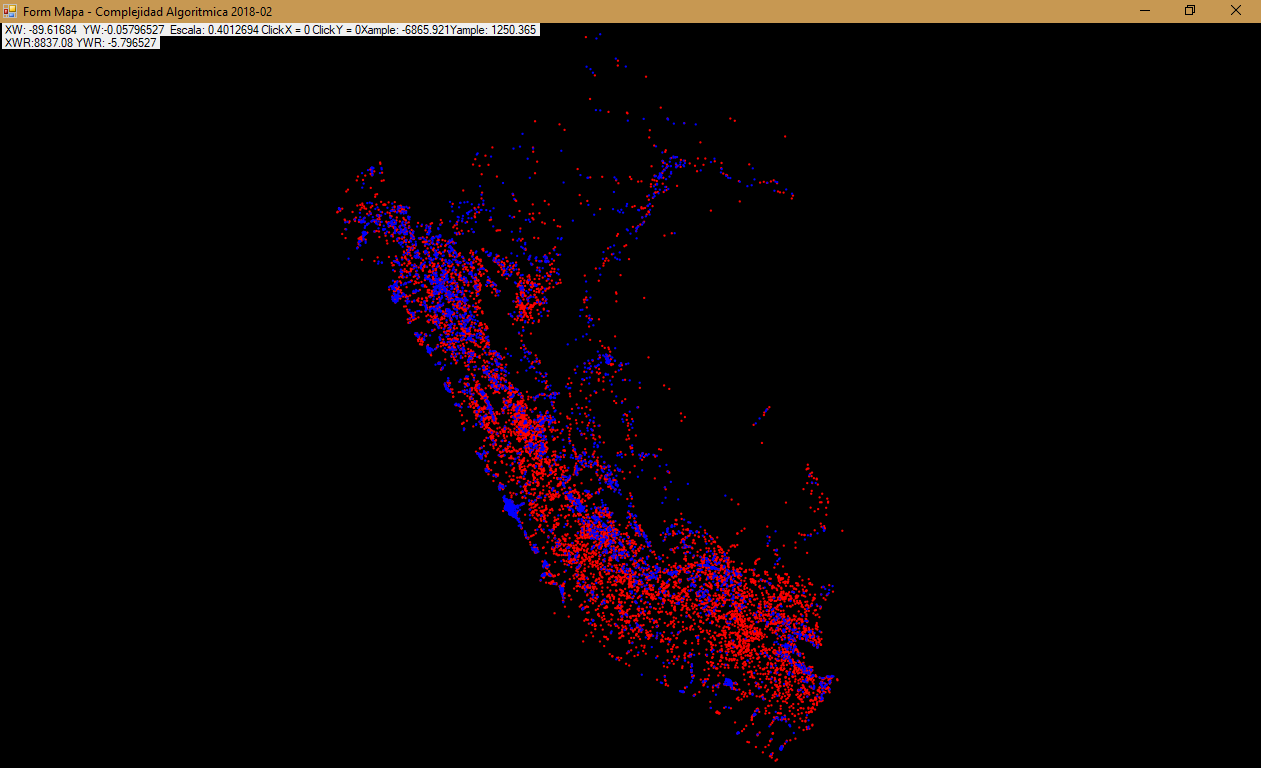
**Rutas cercanas:**

****

**Grafos para aplicar Bellman Ford, Floyd Warshall y Prim**

****

**Grafico: puntos azules centros educativos y puntos rojos centros poblados**

****

* 1. **Diseño de interacción**

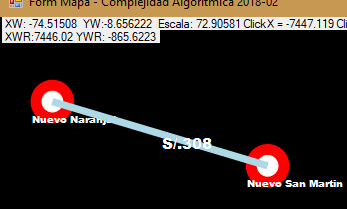
**Para la interacción con la interfaz grafica, se utilizará el mouse y el teclado para la ejecución de algoritmos y pruebas**

**Primero haremos click izquierdo para seleccinar un nodo (por si deseamos deseleccionar hacemos doble click en el nodo)**

****

**Luego hacemos click derecho en el nodo a conectar**

****

****

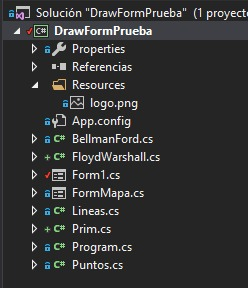
* 1. **Tipos de datos abstractos**

Para nuestro proyecto hemos utilizado los tipos de datos abstractos simples, es decir los que cambian su valor pero no su estructura, tales como

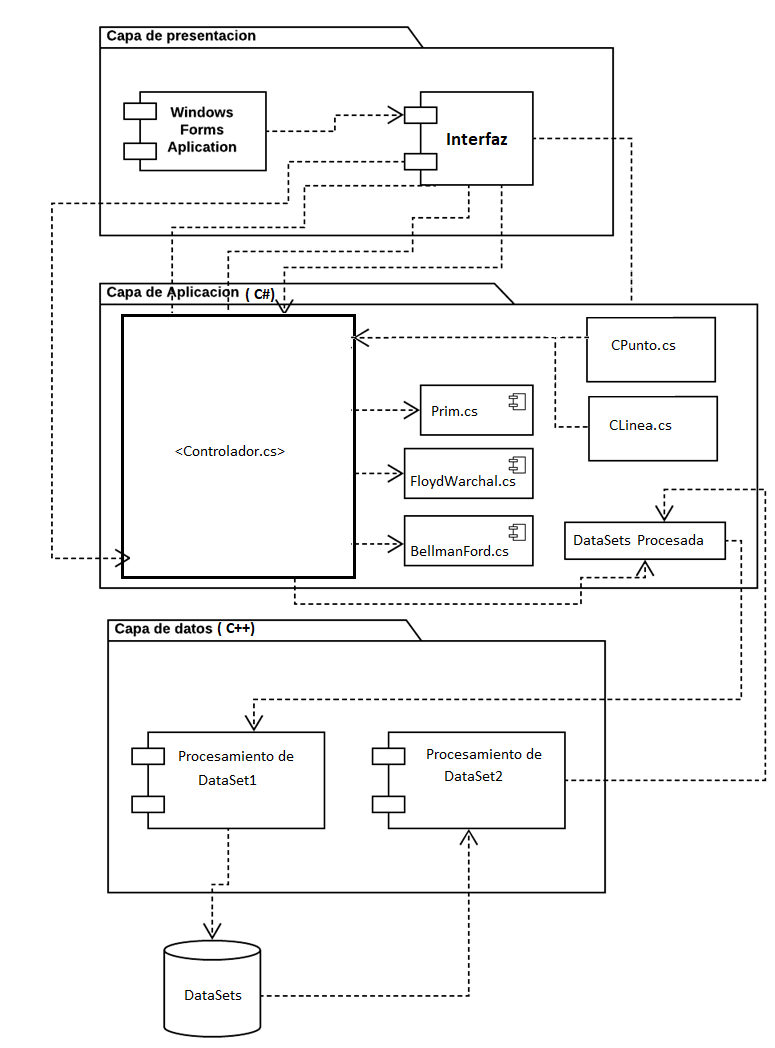
* **enteros**
* **Booleanos**
* **Float**
* **String**
* **Listas**
* **Clases**
* **Diccionarios**

En el caso de los tipos de datos abstractos contenedores, los que si cambian su estructura, hemos utilizado **listas, vectores** para el caso de los grafos se son generados dentro de los algoritmos de Prim, Bellman Ford y Floyd Warshall.

* 1. **Componentes**

****

* 1. **Diagrama de componentes**

****

1. **Bibliografía y anexos**

The Traveling Salesman Problem (NSERC) . (2018). Natural Sciences and Engineering Research Council of Canada. Recuperado de: <http://www.math.uwaterloo.ca/tsp/index.html> [Consulta: 19 de Octubre de 2018].

FloydWarhall Algorithm (2018). Csharp Star © 2018 . Recuperado de: <https://www.csharpstar.com/floyd-warshall-algorithm-csharp/> [Consulta: 10 de Noviembre de 2018].

Prim algorithm (2018). Getting minimum spanning tree using Prim algorithm on C#. Recuperado de <https://gist.github.com/whoo24/c1d3df26f6664ddd9ba9792ad46519c3> [Consulta: : 12 de Noviembre 2018].

Bellman Ford Algorithm (2018). Bellman Ford Algorithm in C#. Recuperado de: <https://gist.github.com/pczajkowski/20af95c37826840f859d7eba5f36e4a0> [Consulta: 15 de Noviembre de 2018].

**Repositorio de GitHub**

Raíz principal

<https://github.com/u201314147/CC76TP20182>

Interfaz grafica con algoritmos de bellman Ford, Floyd Warshall y Prim

<https://github.com/u201314147/CC76TP20182/tree/master/GUI/DrawFormPrueba/DrawFormPrueba>