Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

ENGINEERING DESIGN METHOD

1). **Identificación del problema**

Problema general:

La aerolínea *Fly-Fast* tiene un grave problema, existe un alto grado de insatisfacción por parte de sus clientes debido a las rutas ofrecidas. Los pasajeros afirman que el sistema de escalas ofrecido es inefectivo y afecta negativamente la experiencia de vuelo haciendo que las rutas sean bastantes largas, incómodas y costosas. Esto ha generado que muchos usuarios se movilicen hacia otras aerolíneas debido a que, según ellos, ofrecen un mejor servicio de transporte. El personal de la aerolínea no tiene muy claro cómo solventar está situación, por tal motivo nos ha contratado y pedido una solución tecnológica que permita solucionar este problema.

Síntomas o problemas específicos:

1. **Déficit en evaluar las rutas más cortas**: La aerolínea necesita identificar las rutas más cortas para cada destino, considerando factores como la distancia entre los destinos, las posibles escalas y los tiempos de vuelo.

2. **Incapacidad para identificar las necesidades y preferencias de los usuarios:** La aerolínea debe comprender las necesidades y preferencias de sus usuarios en términos de factores como la duración del vuelo, el costo, la disponibilidad y la conveniencia de los horarios, para poder ofrecer opciones que se ajusten a esas necesidades.

3. **Falta de información de manera clara y accesible**: La aerolínea necesita presentar la información de las rutas más cortas de una manera que sea fácil de entender y acceder para sus clientes, para que puedan tomar decisiones informadas sobre su selección de ruta.

4. **Déficit en evaluación y optimización de la información de las rutas**: La aerolínea debe revisar regularmente la información de las rutas más cortas para asegurarse de que sigue siendo precisa y relevante para los usuarios, y realizar ajustes según sea necesario para mejorar la experiencia de los clientes.

2). Recopilación de información necesaria:

Es pertinente indagar sobre problemas similares y analizar la forma en la cual estos fueron solucionados. A continuación algunas fuentes de información.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

- **"How Delta Air Lines Is Winning Customer Satisfaction With Big Data"**. Este artículo de Forbes habla sobre cómo Delta Air Lines está utilizando la tecnología y el análisis de datos para mejorar la experiencia del cliente. En particular, se menciona la herramienta "Shop by Schedule" que ayuda a los usuarios a encontrar rutas más convenientes.

  Tomado de:
  https://d3.harvard.edu/platform-digit/submission/big-data-takes-flight-at-delta-air-lines/#:~:text=By%20bringing%20data%20to%20what,engage%20customers%20and%20generate%20loyalty.

- **"Emirates Launches AI-Powered Travel Assistant Tool"**. Este artículo de CNBC describe cómo Emirates ha lanzado una herramienta de planificación de viajes basada en inteligencia artificial, que utiliza datos de viaje y preferencias personales para ofrecer recomendaciones de rutas y tarifas a los usuarios.

  Tomado de:

  https://www.marketingdive.com/news/emirates-vacations-digital-display-ads-integrate-chatbots-that-give-travel/518243/

- **"How United Airlines Uses AI to Improve the Customer Experience"**. En este artículo de VentureBeat se describe cómo United Airlines está utilizando la inteligencia artificial para mejorar la experiencia del cliente en todo el viaje, desde la planificación hasta la llegada al destino. Se menciona una herramienta de planificación de viajes llamada "ConnectionSaver" que ayuda a los usuarios a encontrar rutas más convenientes en caso de retrasos o cancelaciones de vuelos.

  Tomado de:

  https://www.cio.com/article/419642/united-airlines-gives-employees-the-digital-tools-to-make-customers-happy.html

Además de la información encontrada, existen varios conceptos que pueden ayudarnos en la comprensión y el análisis de la problemática, tales como:

- **Escala [En vuelo]** Los vuelos con escala, son aquellos en los que la aeronave siempre realiza un desvió en una ciudad antes de aterrizar en el destino final, esa parada es realizada para los pasajeros que van a desembarcar en ese destino, y para que embarquen los que se dirigen al destino final.
  https://www.egali.com.co/blog/vuelos-con-escala/

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

- **Atención al cliente**: La satisfacción es una medida de cómo los productos suministrados y los servicios prestados por una empresa, cumplen o superan las expectativas del cliente.
  https://www.itaerea.es/atencion-cliente-aeropuertos

- **Ruta:** Una ruta es el camino o trayecto que se toma para llegar de un lugar a otro. En el contexto de las aerolíneas, una ruta se refiere a la secuencia de destinos que se visitan en un vuelo.

  https://www.beetrack.com/es/blog/ruta-de-transporte-dise%C3%B1arla

- **Experiencia del usuario:** La experiencia del usuario se refiere a la percepción que tiene un usuario al interactuar con un producto o servicio. En el contexto de las aerolíneas, la experiencia del usuario puede estar influenciada por factores como la facilidad de reserva de vuelos, la calidad del servicio a bordo y la información clara y precisa sobre las rutas y horarios.

  https://www.questionpro.com/blog/es/importancia-de-la-experiencia-del-usuario/

- **Planificación de viajes:** La planificación de viajes se refiere al proceso de organizar y reservar los detalles de un viaje, incluyendo los destinos, las fechas, los horarios, los alojamientos y las actividades. En el contexto de las aerolíneas, la planificación de viajes puede ser un proceso complejo que requiere información precisa y actualizada sobre las rutas y opciones de vuelo.
  https://www.euskadi.eus/contenidos/informacion/8005/eu_2594/adjuntos/docu5_04.pdf

Definiciones de los algoritmos:

- **Dikstra:** El algoritmo de Dijkstra utiliza una estrategia conocida como "búsqueda de costo uniforme" para explorar gradualmente los nodos del grafo y determinar el camino más corto desde un nodo de origen dado hacia todos los demás nodos alcanzables. El algoritmo es muy eficiente y se utiliza ampliamente en aplicaciones de redes y transporte, como enrutamiento de paquetes en Internet y sistemas de navegación.

- **Floyd - Warshall:** A diferencia del algoritmo de Dijkstra, que encuentra el camino más corto entre un par de nodos específicos, el algoritmo de Floyd-Warshall encuentra el camino más corto entre todos los pares de nodos en el grafo. Esto significa que se puede utilizar para calcular las distancias más cortas entre todos los nodos en una sola ejecución del algoritmo, lo que lo hace eficiente para problemas donde se necesita conocer las distancias mínimas entre todos los nodos.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

- **BFS:** Es un algoritmo de búsqueda utilizado para explorar o recorrer estructuras de datos, como grafos o árboles. Se caracteriza por recorrer los nodos de manera horizontal, explorando primero todos los nodos vecinos antes de pasar a los nodos siguientes. El algoritmo BFS comienza en un nodo de origen y visita todos sus nodos vecinos antes de avanzar al siguiente nivel de nodos. Luego, explora todos los vecinos de los nodos recién visitados y continúa este proceso hasta que se hayan visitado todos los nodos alcanzables desde el nodo de origen o se haya encontrado el nodo objetivo (si se está buscando un objetivo específico).

- **Prim:** El objetivo del algoritmo de Prim es encontrar un subconjunto de aristas que forme un árbol que incluya todos los nodos del grafo y tenga el peso total mínimo. Este árbol se conoce como árbol de expansión mínima o árbol generador mínimo.

- **Kruskal:** El objetivo del algoritmo de Kruskal es encontrar un subconjunto de aristas que forme un árbol que incluya todos los nodos del grafo y tenga el peso total mínimo. Este árbol se conoce como árbol de expansión mínima o árbol generador mínimo.

3) Búsqueda de soluciones creativas.

Teniendo en cuenta los algoritmos de grafos que vimos en clase. Estas son algunas de las soluciones posibles para atacar el problema de minimizar el tiempo de viaje y tener el costo mínimo, dependiendo de la necesidad del cliente.

1. **Dijkstra y Floyd- Warshall** : Para utilizar el algoritmo de Dijkstra, debemos construir un grafo que represente todas las rutas posibles entre los destinos ofrecidos por Fly Fast. Asignaremos pesos a las aristas que indiquen el costo o la duración de cada ruta. Luego, ejecutamos el algoritmo de Dijkstra desde el aeropuerto de origen deseado para encontrar el camino más corto hacia todos los demás aeropuertos. Esto nos dará la información necesaria para seleccionar las rutas más eficientes en términos de tiempo y costo.En el caso del algoritmo de Floyd-Warshall, construimos una matriz de adyacencia que representa todas las rutas posibles entre los destinos ofrecidos por Fly Fast. Asignamos pesos a las celdas de la matriz que indican el costo o la duración de cada ruta. Luego, ejecutamos el algoritmo de Floyd-Warshall en la matriz para encontrar la distancia más corta entre todos los pares de aeropuertos. Esto nos permitirá seleccionar las rutas óptimas en términos de tiempo y costo para los pasajeros.

2. **Dijkstra y BFS:** Aplicar el algoritmo de Dijkstra para encontrar la ruta más corta desde el origen hasta el objetivo, retornando una lista que representan, en orden , los vértices a llegar. Esto implica asignar pesos o distancias a las aristas del grafo y calcular las distancias más cortas utilizando el algoritmo de Dijkstra. Utilizar los resultados obtenidos del paso anterior para identificar las rutas más cortas y más

eficientes entre los destinos. Ya con la lista de aeropuertos que representa la ruta más rápida se calculará la distancia de todo el viaje con el algoritmo BFS.

3. **Prim y Kruskal** : El algoritmo de Prim se utiliza para encontrar un árbol de expansión mínima en un grafo ponderado y no dirigido. En este caso, se podría construir un grafo donde los nodos representan los aeropuertos y las aristas representan las posibles rutas entre ellos. Asigna pesos a las aristas según el costo o la duración del viaje. Luego, ejecuta el algoritmo de Prim para encontrar el árbol de expansión mínima que conecte todos los aeropuertos. Esto te dará un conjunto de rutas eficientes y rentables para los pasajeros de Fly Fast. Por otro lado, el algoritmo de Kruskal también se utiliza para encontrar un árbol de expansión mínima en un grafo ponderado y no dirigido. Siguiendo un enfoque similar, podemos construir un grafo donde los nodos sean los aeropuertos y las aristas representan las rutas entre ellos con sus respectivos pesos. Ejecuta el algoritmo de Kruskal para seleccionar un subconjunto de aristas que forman un árbol de expansión mínima que conecte todos los aeropuertos. Este árbol te proporcionará las rutas más eficientes y rentables para los pasajeros de Fly Fast.

4). Transición de ideas a diseños preliminares.

Descartada la alternativa #3.
Sabiendo las opciones disponibles para darle solución al problema, se descarta la opción de implementar **Prim y Kruskal**, ya que aunque los algoritmos de Prim y Kruskal son eficaces para encontrar rutas eficientes y rentables en problemas de selección de rutas, existen algunas consideraciones que podrían dificultar su aplicabilidad en el caso de Fly Fast. Estos algoritmos no tienen en cuenta restricciones adicionales como horarios de vuelo o conexiones entre rutas, lo que limita su capacidad para generar soluciones prácticas en un contexto dinámico y en tiempo real. Para abordar estos desafíos, se requeriría un enfoque adicional para incorporar la información y las restricciones específicas de Fly Fast, a fin de ofrecer rutas adecuadas y satisfacer las necesidades de los pasajeros de manera más precisa y eficiente. En la parte de implementación los algoritmos de Prim y Kruskal pueden presentar dificultades. Estas dificultades incluyen el manejo de estructuras de datos complejas, la optimización del rendimiento en grafos grandes, la adaptación a casos especiales y el manejo adecuado de actualizaciones en tiempo real. Estos desafíos requieren un nivel adicional de conocimiento y experiencia para implementar los algoritmos de manera efectiva.

Lo cual nos deja con las alternativas 1 y 2, las cuales ayudan por las siguientes razones:

Posibles opciones:

Alternativa #1 Dijkstra y Floyd- Warshall

La combinación de los algoritmos de Dijkstra y Floyd-Warshall ofrece una solución integral.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

El algoritmo de Dijkstra es útil para encontrar el camino más corto entre dos aeropuertos específicos. Podemos aplicar este algoritmo para determinar la ruta más eficiente desde un aeropuerto de origen hasta todos los demás aeropuertos. Esto nos ayudará a seleccionar las rutas que minimicen el tiempo total de viaje y eviten escalas innecesarias. Dijkstra se enfoca en un solo origen, lo que permite analizar las rutas de manera individual y garantizar la optimización del tiempo.

Por otro lado, el algoritmo de Floyd-Warshall se utiliza para encontrar la distancia más corta entre todos los pares de aeropuertos en un solo paso. Este algoritmo considera todos los posibles caminos entre los aeropuertos y proporciona la información necesaria para evaluar las rutas más rentables en términos de costos. Floyd-Warshall tiene una visión global y nos permite considerar todas las posibles combinaciones de origen y destino, lo que nos ayuda a seleccionar las rutas más económicas y evitar opciones costosas.

Combinando ambos algoritmos, podemos obtener una solución completa al problema de selección de rutas adecuadas. Podemos utilizar el algoritmo de Floyd-Warshall para analizar todas las combinaciones de origen y destino, obteniendo la distancia más corta y el costo asociado entre cada par de aeropuertos. Luego, podemos utilizar el algoritmo de Dijkstra para encontrar la ruta más corta desde el aeropuerto de origen hacia todos los demás aeropuertos. Esta combinación nos permite identificar las rutas óptimas en términos de tiempo y costo, asegurando una selección eficiente y rentable de rutas para los pasajeros

En resumen, la combinación de los algoritmos de Dijkstra y Floyd-Warshall nos proporciona una solución integral para el problema de selección de rutas de Fly Fast. Dijkstra nos ayuda a optimizar el tiempo de viaje, evitando escalas innecesarias, mientras que Floyd-Warshall nos permite seleccionar las rutas más económicas en términos de costo. Esta combinación garantiza una selección de rutas eficiente y rentable.

Alternativa #2 Dijsktra y BFS

Se puede combinar los algoritmos de Dijkstra y BFS de la siguiente manera para resolver el problema:

Aplicar el algoritmo de Dijkstra para encontrar la ruta más corta desde el origen hasta el objetivo. Esto implica asignar pesos o distancias a las aristas del grafo y calcular las distancias más cortas utilizando el algoritmo de Dijkstra.

Utilizar los resultados obtenidos del paso anterior para identificar las rutas más cortas y más eficientes entre los destinos. Se puede presentar esta rutas como opciones para los usuarios, considerando factores como la duración del vuelo, el costo y la conveniencia de los horarios.

Utilizar el BFS para calcular la distancia total, con la lista de aeropuertos sacadas del Dijkstra que muestra la ruta mas eficiente a ese factor

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Es importante tener en cuenta que la construcción del grafo puede ser costosa computacionalmente si hay muchos destinos y conexiones. En ese caso, podemos aplicar optimizaciones como limitar la exploración a un número máximo de escalas o utilizar heurísticas para guiar la búsqueda y reducir el espacio de exploración.

5) Evaluación y selección de la mejor solución:

En el proceso de evaluación y selección se va a tomar en cuenta los criterios de:
facilidad de implementación, eficiencia del algoritmo en búsqueda de mejores rutas, eficiencia espacial,eficiencia temporal y flexibilidad y adaptabilidad.

Sistema de criterios y calificaciones:

Criterio A. facilidad de implementación:
[1] Es muy complejo implementar el algoritmo.
[2] No es ni muy complejo ni muy fácil implementar el algoritmo.
[1] Es sencillo implementar el algoritmo

Criterio B. Eficiencia del algoritmo.

[1] Los algoritmos son propensos a fallos fáciles de evitar.
[2] Los algoritmos son propensos a fallos que se provocan.
[3] Los algoritmos casi nunca fallan en casi ninguna situación.

Criterio C. Eficiencia espacial.

[1] El algoritmo requiere mucho espacio en memoria.
[2] El algoritmo requiere el espacio necesario en memoria.
[3] El algoritmo requiere muy poco espacio en memoria.

Criterio D. Eficiencia temporal.

[1] El algoritmo requiere mucho tiempo de ejecución en función del tamaño de entrada.
[2] El algoritmo requiere el tiempo necesario de ejecución en función de la entrada.
[3] El algoritmo requiere poco tiempo de ejecución en función del tamaño de entrada.

Criterio E. Flexibilidad y adaptabilidad.
[1] El algoritmo es muy poco flexible y adaptable.
[2] El algoritmo es medianamente flexible y adaptable.
[3] El algoritmo es muy flexible y adaptable

Sistema de evaluación:

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

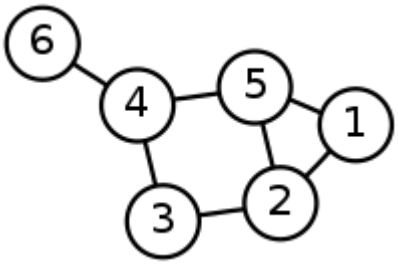
	Criterio A	Criterio B	Criterio C	Criterio D	Criterio E	Total
Dijkstra y Floyd-Warshall	1	2	3	2	2	10
Dijsktra y BFS	3	1	3	1	3	11

Selección:

La alternativa 2, que utiliza los algoritmos de Dijkstra y BFS, se destaca como la mejor opción. Es fácil de implementar, eficiente en términos de tiempo y espacio, y flexible en su aplicación. Dijkstra encuentra rutas cortas entre dos nodos, mientras que BFS es útil para recorrer grafos de manera exhaustiva. Es decir, esta alternativa cumple con todos los criterios evaluados, lo que la convierte en la elección ideal para resolver el problema planteado.

TAD´S

TAD Grafo



Abstract Object: Grafo
Grafo={Vertexes = <vertexes>, Edges<edges>, isManaged=boolean}

{Inv:
***vertexes > 0**
***if (isManaged) a edge b =! b edge a**
}

Primitive Operations:
Modifiers:

- **AddVertex: Graph X Vertex -> Graph**
- **AddEdge: Graph X Vertex1 X Vertex2 X Weight -> Graph**
- **DeleteVertex: Graph X (T valueVertex) ->Graph**

Analyzers:

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

- **BFS: Graph X VertexSource -> ArrayList:**
- **DFS: Graph X VertexSource -> ArrayList:**
- **Dijkstra: Graph X VertexSource -> int (minimum distance between source and other vertex)**
- **Floyd-Warshall:**

AddVertex
It adds a new vertex in the graph.
{*pre*: {{Graph:...} ∧ v} ∧ v ∈ Vertex}
{*post*: Graph.vertexes = new v}

AddEdge
It adds an edge between two vertexes.
{*pre*: {Graph…}, {v1 ≠Nil, v2 ≠Nil}, {e between v1 ∧ v2 = Nil } ∧ v1, v2 ∈ Vertex ∧ e ∈ Edge}
{*post*: Graph.edges = new e }

DeleteVertex
It deletes a vertex of the graph.
{*pre*:{{Graph:...} ∧ v} ∧ v ∈ Vertex}
{*post*: *remove* Graph.vertexes.v}

BFS
It explores a graph starting on a vertex and carries on with all its neighbors.
{*pre*: :{{Graph:...} ∧ vSource} ∧ vSource ∈ Vertex}
{*post*:{BF tree}}

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

DFS
It explores all the graph starting in the first vertex
{*pre*: :{{Graph:...} ∧ vSource} ∧ vSource ∈ Vertex}
{*post*:{ DF forest}}

Dijkstra
It returns the shortest path between a vertex and another.
{*pre*: :{{Graph:...} ∧ vSource} ∧ vSource ∈ Vertex}
{*post*: Path between v ∧ v' with less weight}

Floyd-Warshall
It returns the shortest path between all the pairs of vertexes.
{*pre*:{{Graph:...} ∧ vSource} ∧ vSource ∈ Vertex}
{*post*: Path between v ∧ the others vertexes with the less weight }

THE PROBLEM TO SOLVE

1). Situación problemática:

La aerolínea Fly Fast nos ha contratado para darle fin a una problemática que les ha afectado su reputación, su economía y su imagen hacia sus clientes fieles. El problema que enfrenta Fly Fast con respecto a la selección de rutas adecuadas es un asunto de gran importancia. Esta aerolínea se dedica a brindar un servicio eficiente y de alta calidad a sus usuarios, y la selección de rutas es un aspecto crítico en la satisfacción del cliente, pues el cliente es nuestra principal fuente de ingresos.

El problema parte que varios empleados de la aerolínea no conocen ni distinguen todas las múltiples rutas que hay disponibles y esto provoca que no se hayan seleccionado rutas adecuadas , esto puede afectar a los pasajeros en varios aspectos. En primer lugar, si se

seleccionan rutas con escalas innecesarias, se puede aumentar el tiempo total de viaje de los pasajeros y causarles molestias innecesarias. En segundo lugar, si se seleccionan rutas costosas, esto puede afectar negativamente a los clientes que buscan ahorrar dinero y pueden optar por otras aerolíneas.

Fly Fast se ha comprometido a mejorar continuamente su servicio y garantizar que sus usuarios reciban la mejor atención posible. Es por eso que nos han contratado para desarrollar un sistema que pueda proporcionar rutas eficientes y rentables para nuestros pasajeros.

Esta problemática puede afectar la reputación y la rentabilidad de Fly Fast, ya que los usuarios pueden optar por otras opciones de viaje si no están satisfechos con las opciones de rutas ofrecidas. Es por eso que es crucial encontrar una solución efectiva y eficiente a este problema.

2) Análisis del problema y de los requerimientos funcionales.

Client	Airline Fly Fast
User	*Fly-Fast* Tripulation
Functional requirements	R1. Permitir al usuario ingresar nuevas ciudades destino R2. Permitir al usuario ingresar nuevas conexiones o rutas entre ciudades R3. Permitir al usuario elegir su prioridad entre tiempo o costo. R4. Mostrar al usuario la ruta que lleva menos tiempo R5. Mostrar al usuario la ruta la ruta que tiene un menor precio.
Context of the problem	El problema parte que varios empleados de la aerolínea no conocen ni distinguen todas las múltiples rutas que hay disponibles y esto provoca que no se hayan seleccionado rutas adecuadas , esto puede afectar a los pasajeros en varios aspectos. En primer lugar, si se seleccionan rutas con escalas innecesarias, se puede aumentar el tiempo total de viaje de los

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

	pasajeros y causarles molestias innecesarias. En segundo lugar, si se seleccionan rutas costosas, esto puede afectar negativamente a los clientes que buscan ahorrar dinero y pueden optar por otras aerolíneas.

R1.

Nombre o identificador	Permitir al usuario ingresar nuevas ciudades destino		
Resumen	Permite agregar un nuevo vértice al grafo. El vértice representa una ciudad de destino.		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición de selección o repetición**
	city	String	que el dato sea diferente a null
Actividades generales necesarias para obtener los resultados	Los datos requeridos deben ser válidos.		
Resultado o Postcondición	Creación exitosa del vértice.		

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Salidas	**Nombre salida**	**Tipo de dato**	**Condición de selección o repetición**
	msj	String	

R2.

Nombre o identificador	Permitir al usuario ingresar nuevas conexiones o rutas entre ciudades		
Resumen	Este requerimiento permite al sistema crear conexiones entre dos ciudades ya existentes.		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición de selección o repetición**
	city1	City	que la ciudad exista
	city2	City	que la ciudad exista

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

	distance	double	que no sea null o diferente a un tipo double
	time	LocalTime	que no sea null o diferente a tipo LocalTime
Actividades generales necesarias para obtener los resultados	que todos los datos sean rellenados de forma correcta para evitar excepciones o resultados no esperados.		
Resultado o Postcondición	la creación de una nueva arista que simboliza el viaje de una ciudad a otra		
Salidas	**Nombre salida**	**Tipo de dato**	**Condición de selección o repetición**
	msj	String	

R3.

Nombre o identificador	Permitir al usuario elegir su prioridad entre tiempo o costo.

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

Resumen	Se le pide al usuario saber si prefiere tener mejor tiempo o mejor costo para su vuelo, así saber qué alternativa le parecería mejor.		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición de selección o repetición**
	option	int	que sea de tipo int, diferente a null y representa una opción válida.
	city1	City	que sea una ciudad que exista en el sistema
	city2	City	que sea una ciudad que exista en el sistema
Actividades generales necesarias para obtener los resultados	El usuario deberá marcar una opción válida para dar solución a su problema.		
Resultado o Postcondición	Se le mostrará al usuario el camino ideal de acuerdo a sus peticiones.		
Salidas	**Nombre salida**	**Tipo de dato**	**Condición de selección o repetición**
	msj	String	

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

R4.

Nombre o identificador	Mostrar al usuario la ruta que lleva menos tiempo		
Resumen	Si el usuario desea conocer la ruta que le lleva menos tiempo, entonces marca esta opción.		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición de selección o repetición**
Actividades generales necesarias para obtener los resultados	El usuario escogió esta opción, y se llama al método que busca la mejor ruta.		
Resultado o Postcondición	un mensaje que te muestre cual es la mejor ruta, para el viaje solicitado por nuestro cliente.		
Salidas	**Nombre salida**	**Tipo de dato**	**Condición de selección o repetición**
	msj	String	

Santiago Hernández Saavedra A00382538
Juan Jose Barrera A00394876
Sergio Florez Sanabria A00396046

R5.

Nombre o identificador	Mostrar al usuario la ruta la ruta que tiene un menor precio.		
Resumen	el usuario escoge la opción que le muestra la ruta que tiene el menor precio		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición de selección o repetición**
Actividades generales necesarias para obtener los resultados	El usuario escogió esta opción y se le mostrará la ruta más económica para llegar a su destino.		
Resultado o Postcondición	un mensaje que le muestre cual es la ruta más económica para llegar a su destino.		
Salidas	**Nombre salida**	**Tipo de dato**	**Condición de selección o repetición**
	msj	String	