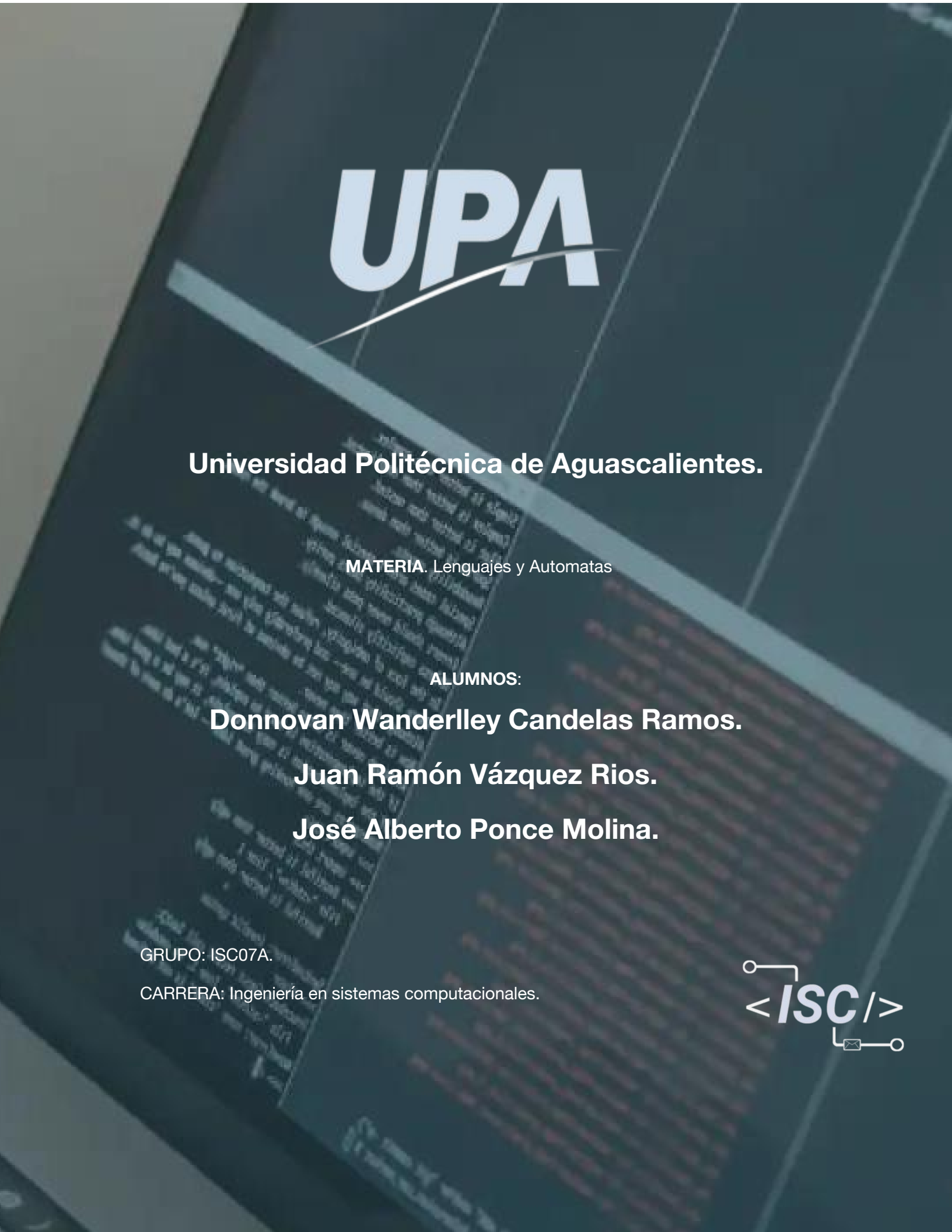


Universidad Politécnica de Aguascalientes.

MATERIA. Lenguajes y Automatas

ALUMNOS:

Donnovan Wanderlley Candelas Ramos.

Juan Ramón Vázquez Rios.

José Alberto Ponce Molina.

GRUPO: ISC07A.

CARRERA: Ingeniería en sistemas computacionales.

Índice

1. Introducción

El presente proyecto consiste en el desarrollo de un sistema basado en grafos no dirigidos, cuyo propósito es modelar relaciones bidireccionales entre distintos nodos. Este tipo de grafos permite representar situaciones en las que la conexión entre dos elementos funciona en ambos sentidos, como rutas, caminos o enlaces equivalentes.

El sistema implementa el algoritmo de Dijkstra para calcular los caminos más óptimos dentro del grafo, permitiendo analizar distancias y seleccionar rutas eficientes. Además, cuenta con una interfaz interactiva desarrollada con componentes JSX, lo que facilita la visualización de la información y la interacción del usuario durante la ejecución del programa.

2. Tipo de Grafo

2.1 Descripción formal del grafo

El grafo implementado en el proyecto es un grafo no dirigido, lo que significa que las aristas no poseen una orientación específica. Si existe una conexión entre dos nodos, esta puede recorrerse en ambos sentidos, manteniendo el mismo peso o costo en cada dirección.

De manera formal, el grafo puede definirse como (G = (V, E)), donde:

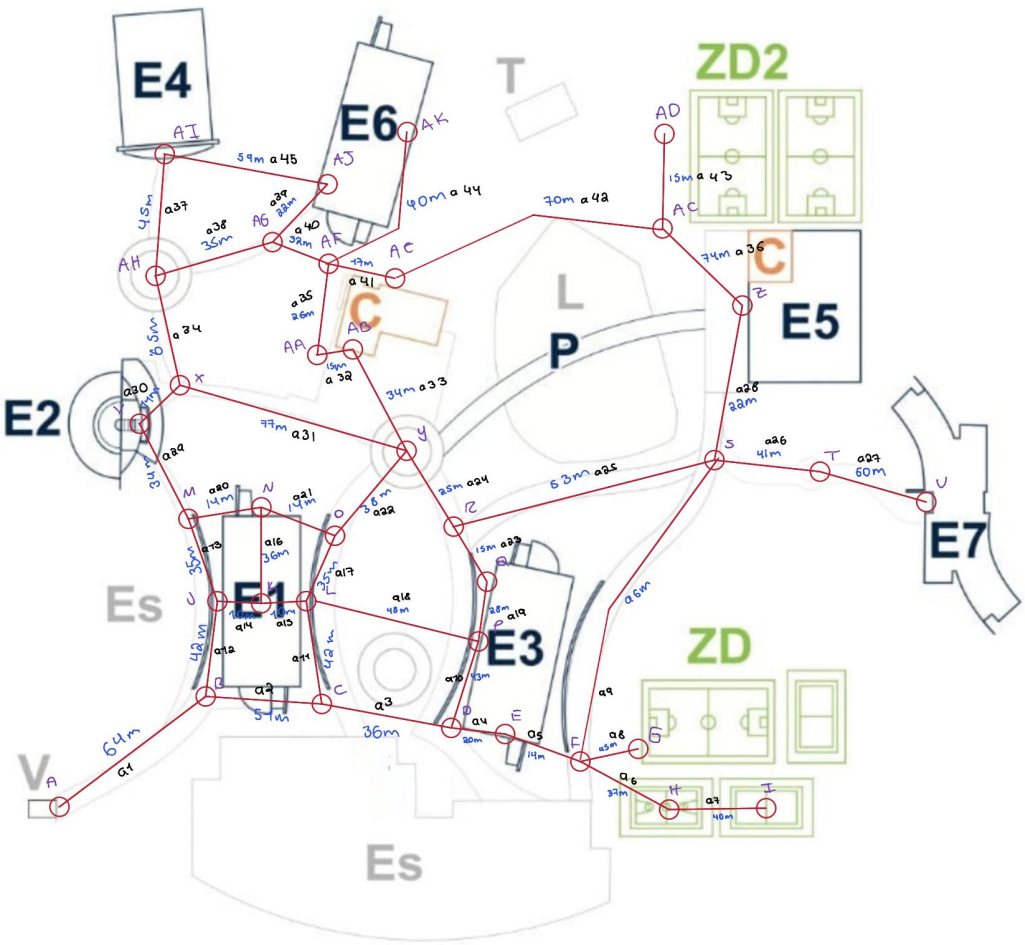
(V) representa el conjunto de nodos o vértices del grafo.

(E) representa el conjunto de aristas, las cuales establecen las conexiones entre los nodos.

Cada nodo simboliza un punto dentro del sistema, mientras que las aristas representan las relaciones existentes entre ellos. Al tratarse de un grafo no dirigido, la relación entre dos nodos es bidireccional, permitiendo el tránsito en ambos sentidos sin restricciones.

Además, el grafo es ponderado, ya que cada arista tiene asignado un valor numérico que representa el costo o distancia entre los nodos conectados. Estos pesos son utilizados por el algoritmo de Dijkstra para calcular las rutas más óptimas dentro del sistema.

Este tipo de grafo es adecuado para representar escenarios donde las relaciones son mutuas, como caminos entre puntos o conexiones físicas, ya que refleja de manera precisa la igualdad de condiciones en ambos sentidos de la conexión.



V = {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, X, Y, Z, AA, AB, AC, AD, AE, AF, AG, AH, AI, AJ, AK}

a1 = (A, B, 64 m)

a2 = (B, C, 51 m)

a3 = (B, U, 42 m)

a4 = (C, O, 36 m)

a5 = (C, L, 42 m)

a6 = (D, P, 43 m)

a7 = (D, E, 20 m)

a8 = (E, F, 14 m)

a9 = (F, G, 45 m)

a10 = (F, S, 96 m)

a11 = (F, H, 37 m)

a12 = (H, I, 40 m)

a13 = (J, M, 35 m)

a14 = (J, K, 10 m)

a15 = (K, L, 10 m)

a16 = (K, N, 36 m)

a17 = (L, P, 48 m)

a18 = (L, O, 35 m)

a19 = (M, N, 14 m)

a20 = (M, V, 34 m)

a21 = (N, O, 14 m)

a22 = (O, Y, 38 m)

a23 = (P, Q, 28 m)

a24 = (Q, R, 15 m)

a25 = (R, Y, 25 m)

a26 = (R, S, 53 m)

a27 = (S, T, 41 m)

a28 = (S, Z, 22 m)

a29 = (T, V, 50 m)

a30 = (V, X, 11 m)

a31 = (X, Y, 77 m)

a32 = (X, AH, 65 m)

a33 = (Y, AB, 34 m)

a34 = (Z, AC, 74 m)

a35 = (AA, AB, 15 m)

a36 = (AA, AF, 26 m)

a37 = (AC, AD, 15 m)

a38 = (AC, AE, 70 m)

a39 = (AE, AF, 17 m)

a40 = (AF, AG, 32 m)

a41 = (AF, AK, 40 m)

a42 = (AG, AJ, 22 m)

a43 = (AG, AH, 35 m)

a44 = (AH, AI, 45 m)

a45 = (AI, AJ, 59 m)

3. Arquitectura General del Proyecto

El proyecto se encuentra organizado en distintos directorios, cada uno con un peso específico, lo que permite mantener una estructura clara y modular. La separación de archivos facilita el mantenimiento del codigo, la escalabilidad del sistema y la comprensión general del funcionamiento del proyecto.

4. Estructura de Directorios

4.1 Directorio /css

El directorio /css almacena el archivo de estilos del proyecto. En este archivo se definen las configuraciones visuales que controlan la apariencia de la interfaz, como colores, tamaños, distribución de elementos y estilos generales. Su función principal es mejorar la experiencia del usuario, garantizando una presentación visual clara y coherente durante la ejecución del sistema.

4.2 Directorio /events

El directorio /events contiene un archivo que implementa una función basada en el algoritmo de Dijkstra. Esta función se encarga de calcular los caminos más óptimos dentro del grafo, evaluando las distancias entre los nodos y seleccionando la ruta de menor costo. El algoritmo de Dijkstra es utilizado debido a su eficiencia para encontrar el camino más corto en grafos ponderados, lo que lo hace ideal para el tipo de problema que aborda este proyecto.}

4.3 Directorio /components

El directorio /components almacena tres componentes JSX que funcionan como vistas del sistema. Estos componentes permiten generar interactividad y mostrar información relevante al usuario en tiempo real. Cada componente cumple una función específica dentro de la interfaz, facilitando la visualización del grafo, la interacción del usuario con el sistema y la presentación de resultados obtenidos a partir del algoritmo de Dijkstra.

5. Nodos del Grafo

Dentro de la arquitectura del proyecto, los nodos representan los elementos fundamentales del grafo no dirigido sobre el cual opera el sistema. Cada nodo corresponde a un punto específico dentro de la estructura y sirve como base para establecer conexiones con otros nodos.

Estos nodos son utilizados por el sistema para construir el grafo que posteriormente es procesado por el algoritmo de Dijkstra, ubicado en el directorio /events, y visualizado a través de los componentes JSX almacenados en /components.

5.1 Definición de los Nodos

Un nodo puede entenderse como una entidad individual dentro del grafo. Cada uno cuenta con una identidad única que permite distinguirlo de los demás y facilita la creación de relaciones dentro del sistema. Los nodos no funcionan de manera aislada, sino que forman parte de una red estructurada que representa las conexiones del problema planteado en el proyecto.

5.2 Relación entre Nodos en un Grafo No Dirigido

Debido a que el grafo implementado es no dirigido, las conexiones entre los nodos no tienen un sentido único. Esto significa que cuando dos nodos están conectados, el recorrido entre ellos puede realizarse en ambos sentidos. Esta característica es esencial para el funcionamiento del sistema, ya que permite que el algoritmo de Dijkstra evalúe todas las posibles rutas sin restricciones de dirección, garantizando resultados óptimos en el cálculo de caminos.

5.3 Uso de los Nodos dentro del Sistema

Los nodos son utilizados directamente por la lógica del proyecto para: construir la estructura del grafo, establecer conexiones bidireccionales con otros nodos, servir como puntos de inicio y destino en el cálculo de rutas óptimas y mostrar información relevante al usuario a través de los componentes interactivos.

6. Aristas del Grafo

Las aristas representan las conexiones entre los nodos del grafo y son las encargadas de definir cómo se relacionan entre sí los distintos elementos del sistema. En conjunto con los nodos, las aristas permiten construir la estructura completa del grafo no dirigido utilizado en el proyecto.

6.1 Función de las Aristas

La función principal de las aristas es conectar los nodos y permitir el desplazamiento entre ellos. A través de estas conexiones, el sistema puede representar relaciones entre distintos nodos, definir rutas posibles dentro del grafo y calcular caminos óptimos utilizando el algoritmo de Dijkstra.

6.2 Aristas en un Grafo No Dirigido

Dado que el grafo implementado es no dirigido, las aristas no tienen una orientación específica. Esto significa que una conexión entre dos nodos puede recorrerse en ambos sentidos sin ninguna restricción. Esta característica garantiza que las rutas sean evaluadas de manera equitativa desde cualquier nodo de origen hacia un nodo destino, lo cual es fundamental para el correcto funcionamiento del algoritmo de Dijkstra dentro del sistema.

6.3 Peso de las Aristas

Cada arista puede contar con un peso o costo asociado, el cual representa la distancia, el tiempo o cualquier valor relevante entre dos nodos conectados. Estos pesos son utilizados por el algoritmo de Dijkstra para determinar cuál es el camino más óptimo dentro del grafo. La correcta asignación de los pesos en las aristas permite que los resultados obtenidos reflejen de manera precisa las condiciones del problema planteado.

6.4 Importancia de las Aristas en el Proyecto

Las aristas son un elemento clave dentro del proyecto, ya que permiten establecer relaciones reales entre los nodos, evaluar diferentes rutas posibles y obtener resultados óptimos en el cálculo de caminos. En conjunto con los nodos, las aristas conforman la base estructural del sistema, haciendo posible el análisis y la visualización de las relaciones dentro del grafo.

7. Algoritmo de Dijkstra

El algoritmo de Dijkstra es el encargado de calcular los caminos más óptimos dentro del grafo no dirigido implementado en el proyecto. Su función principal es determinar la ruta de menor costo entre un nodo de origen y los demás nodos del grafo, tomando en cuenta los pesos asignados a las aristas. En este sistema, la lógica del algoritmo se encuentra implementada dentro del directorio /events, lo que permite separar el procesamiento de los datos de la interfaz y mantener una estructura organizada del proyecto.

7.1 Función del Algoritmo dentro del Sistema

El algoritmo de Dijkstra se utiliza para analizar la estructura del grafo construida a partir de nodos y aristas, evaluando todas las rutas posibles entre los puntos conectados. Su función principal dentro del sistema es: recibir un nodo de inicio, analizar las conexiones disponibles desde ese nodo, calcular la distancia mínima hacia los demás nodos y determinar el camino más corto entre dos puntos.

7.2 Funcionamiento General del Algoritmo

El funcionamiento del algoritmo de Dijkstra dentro del proyecto se basa en un proceso iterativo que evalúa progresivamente los nodos del grafo. En cada iteración, el algoritmo selecciona el nodo con la menor distancia conocida y actualiza los valores de sus nodos vecinos. Este procedimiento se repite hasta que todos los nodos han sido evaluados o hasta que se ha encontrado el camino óptimo hacia el nodo destino. Gracias a este enfoque, el algoritmo garantiza que las rutas obtenidas sean las de menor costo posible.

7.3 Relación con Nodos y Aristas

El algoritmo de Dijkstra depende directamente de la correcta definición de los nodos y las aristas del grafo. Los nodos funcionan como los puntos de origen y destino, mientras que las aristas, junto con sus pesos, determinan el costo de desplazamiento entre ellos. Al tratarse de un grafo no dirigido, el algoritmo puede recorrer las aristas en ambos sentidos, evaluando todas las combinaciones posibles de rutas sin restricciones de dirección.

7.4 Importancia del Algoritmo en el Proyecto

El algoritmo de Dijkstra es una parte esencial del proyecto, ya que permite transformar la estructura del grafo en información útil para el usuario. Gracias a su implementación, el sistema puede identificar rutas óptimas dentro del grafo, mostrar resultados claros y

precisos y apoyar la toma de decisiones basada en los caminos calculados. Sin este algoritmo, el proyecto se limitaría únicamente a la representación del grafo, sin la capacidad de realizar análisis de rutas de manera eficiente.

8. Componentes JSX e Interacción con el Usuario

El directorio /components contiene tres componentes JSX que funcionan como vistas interactivas. Estas vistas permiten al usuario: visualizar el grafo, seleccionar nodos de inicio/destino, iniciar el cálculo de rutas y ver los resultados obtenidos por el algoritmo de Dijkstra. Cada componente está diseñado para separar responsabilidades: una vista para la representación gráfica del grafo, otra para los controles/entradas del usuario y otra para mostrar los resultados y detalles de la ruta.

9. Conclusión

Este documento presenta la descripción completa del proyecto: estructura, nodos, aristas, algoritmo de Dijkstra y componentes de la interfaz. La separación modular del proyecto permite mantener el código organizado y facilita futuras mejoras. La inclusión de la imagen del grafo en la ubicación recomendada ayudará al lector a comprender visualmente la estructura y las relaciones tratadas en el proyecto.

10. Referencias

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms* (4th ed.). MIT Press.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*(1), 269–271. https://doi.org/10.1007/BF01386390

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data structures and algorithms in Java* (6th ed.). Wiley.

Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley Professional.

GeeksforGeeks. (s.f.). *Dijkstra's shortest path algorithm*. Recuperado de https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm/

Khan Academy. (s.f.). *Graph representation and traversal*. Recuperado de https://www.khanacademy.org/computing/computer-science/algorithms