UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS

ESCUELA DE INGENIERÍA Y ARQUITECTURA

ESTRUCTURAS DISCRETAS

DISEÑO DE RED DE DISTRIBUCIÓN INTELIGENTE

RAQUEL ARCE ARIAS

JURGUEN FUENTES MADRIGAL

KENNY RAMOS WONG

SAN JOSÉ, AGOSTO 2025

Contents

CAPÍTULO I: JUSTIFICANTES

Objetivos

Objetivo General

Diseñar una red de distribución tolerante a fallos que permita el flujo constante de mercadería en todo el país.

Objetivos Específicos

- Explorar algoritmos de grafos para encontrar las mejores rutas.
- Diseñar un escenario de fallos utilizando tablas de verdad.
- Implementar ciclos de mantenimiento utilizando aritmética modular.
- Buscar rutas alternas ante escenarios críticos.
- Investigar sobre herramientas que nos faciliten el procesamiento de un escenario realista.

Justificación

Aplicar una serie de conocimientos vistos en clase a un escenario que simule un reto al que nos podemos enfrentar en un escenario profesional, diseñado de tal forma que nos sea familiar, pero a la vez retador de la mano de como pasar de comprender términos a aplicarlos con criterio y creatividad en un caso empresarial.

Para el desarrollo del caso planteamos una empresa que debe distribuir productos varios de consumo a nivel nacional, tomando en cuenta que las materias primas ingresan por los puertos y se entregan a distintos centros de distribución que garanticen el inventario en las distintas cedes ubicadas en el país.

Al margen de ello, se aplican distintos conocimientos vistos en clase como lo son el diseño de toda una red conceptual pintando un grafo en el mapa de Costa Rica, aplicando álgebra de Boole para simulación de fallos de una ruta y planeando ciclos de mantenimiento mediante el uso de aritmética modular. Para encontrar las rutas mas optimas utilizaremos distintos algoritmos entre los que se incluyen Disktra, DFS, BFS y de Prim, algoritmos que nos permitirán trazar rutas en el mapa.

Finalmente, se investiga acerca de soluciones tecnológicas que nos permitan plantear un escenario como el descrito, por lo tanto, el proyecto utiliza Neo4j para almacenar los datos, mientras que la librería NetworkX para la ejecución de los algoritmos.

CAPÍTULO II: DESARROLLO

Diseño Conceptual

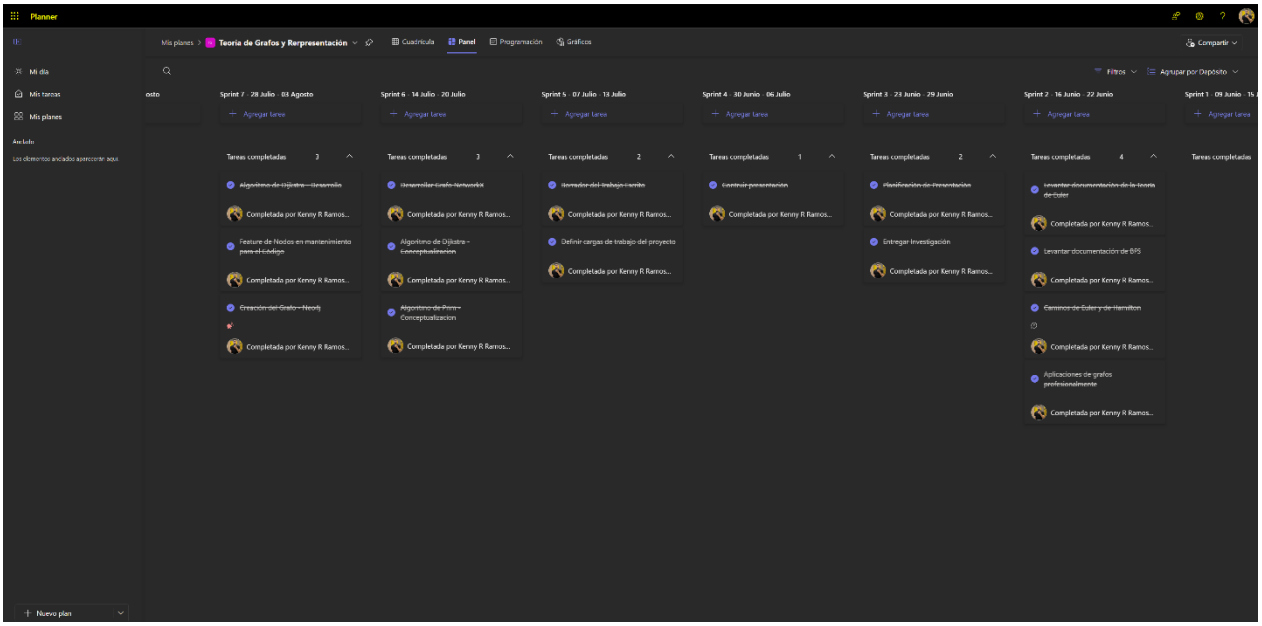
El proyecto parte de una lluvia de ideas en la cuál definimos el escenario, inicialmente como un concepto básico, pero poco abstracto que nos permite dimensionar los alcances del mismo y entender con un escenario familiar como se comportará el grafo, para esta primera etapa, lo que hicimos fue tomar una captura de pantalla en Google Maps, y ubicar las ciudades mas importantes que consideramos para tener operación.

Una vez definidas las ciudades trazamos líneas por las rutas más importantes del país interconectando estas ciudades de tal forma que nos apegáramos a la realidad, por ejemplo, dado que Atenas fue una de estas ciudades y Orotina también, pusimos que Orotina por estar cerca de la pista y ser un cruce para ir a Jacó o Puntarenas fuese un nodo requerido para llegar a estas últimas, mientras que Atenas, al ser un desvío no fuese un requisito para llegar a Orotina, aunque en el mapa parezca que lo es a nivel macro, a nivel micro implica tomar la rampa hacia Atenas y luego devolverse a la 27 para continuar hasta Orotina, por lo tanto, nuestro nodo de San José, tiene una arista que lo conecta a Orotina directamente sin necesidad de parar primero en Atenas.

Fuente: *Elaboración propia*

Metodología de trabajo

Dado que queremos emular un proyecto empresarial realista, decidimos incluir esta parte del desarrollo en el mismo, y es que para dar trazabilidad del trabajo en equipo realizado, utilizamos una herramienta suministrada con el correo universitario llamada Microsoft Planner, que nos permite simular un escenario de trabajo combinando metodologías de Scrum y un tablero Kanban, con entregables definidos a través de distintos Sprint de una semana cada uno.



Fuente: *Elaboración propia*

Mapeo de datos

Una vez conceptualizado el proyecto procedimos a la construcción de los datos a partir de las reglas del escenario planteado, para ello, investigamos sobre distintas formas de visualizar un grafo que sea entendible a nivel de código, pero también de fácil lectura.

Para obtener estos datos, empezamos a conectar cada ciudad mediante el uso de Google Maps y ver cuanta distancia había de un punto a otro, solo hubo un caso de excepción, y fue Limón ya que pusimos a Siquirres como un punto de paso para poder llegar al puerto, pero nos dimos cuenta que este nodo quedaba aislado cuando ni en el escenario real ni a nivel práctico nos iba a ser útil.

Por lo tanto, decidimos conectar una ruta directa mapeando la distancia entre Turrialba (J) y Limón (V) así como otra entre Siquirres (Q) y Limón (V), sin embargo, decidimos no conectarlo directamente con Guápiles ya que las 3 rutas alternas reales serian Turrialba, Zurquí (Pasando por Guápiles) o Vara Blanca. Esta es la única licencia que nos tomamos con las conexiones para que tengan sentido.

Una vez mapeados todos estos datos los escribimos en una representación en JSON para que fuese fácil de comprender y además explorar este tipo de archivos. Para la definición de la metadata de cada nodo:


```
"nodos": {
  "A": {"Provincia": "San José", "Cantón": "San José", "Distrito": "Pavas"},
  "B": {"Provincia": "Heredia", "Cantón": "Heredia", "Distrito": "Ulloa"},
  "C": {"Provincia": "Alajuela", "Cantón": "Alajuela", "Distrito": "Rio Segundo"},
  "D": {"Provincia": "Cartago", "Cantón": "Cartago", "Distrito": "Parte Oriental"},
  "E": {"Provincia": "San José", "Cantón": "Mora", "Distrito": "Colón"},
  "F": {"Provincia": "Limón", "Cantón": "Pococí", "Distrito": "Guápiles"},
  "G": {"Provincia": "Alajuela", "Cantón": "Atenas", "Distrito": "Atenas"},
  "H": {"Provincia": "Alajuela", "Cantón": "San Ramón", "Distrito": "Volio"},
  "I": {"Provincia": "San José", "Cantón": "Puriscal", "Distrito": "Santiago"},
  "J": {"Provincia": "Cartago", "Cantón": "Turrialba", "Distrito": "Suiza"},
  "K": {"Provincia": "Alajuela", "Cantón": "Orotina", "Distrito": "Pozón"},
  "L": {"Provincia": "Puntarenas", "Cantón": "Garabito", "Distrito": "Jacó"},
  "M": {"Provincia": "Puntarenas", "Cantón": "Puntarenas", "Distrito": "Puntarenas"},
  "N": {"Provincia": "Alajuela", "Cantón": "San Carlos", "Distrito": "Quesada"},
  "O": {"Provincia": "Alajuela", "Cantón": "San Carlos", "Distrito": "Fortuna"},
  "P": {"Provincia": "Heredia", "Cantón": "Sarapiquí", "Distrito": "Puerto Viejo"},
  "Q": {"Provincia": "Limón", "Cantón": "Siquirres", "Distrito": "Siquirres"},
  "R": {"Provincia": "San José", "Cantón": "Perez Zeledón", "Distrito": "Daniel Flores"},
  "S": {"Provincia": "Puntarenas", "Cantón": "Aguirre", "Distrito": "Quepos"},
  "T": {"Provincia": "Puntarenas", "Cantón": "Osa", "Distrito": "Uvita"},
  "U": {"Provincia": "Puntarenas", "Cantón": "Buenos Aires", "Distrito": "Buenos Aires"},
  "V": {"Provincia": "Limón", "Cantón": "Limón", "Distrito": "Limón"},
  "W": {"Provincia": "Guanacaste", "Cantón": "Cañas", "Distrito": "Cañas"},
  "X": {"Provincia": "Guanacaste", "Cantón": "Liberia", "Distrito": "Liberia"},
  "Y": {"Provincia": "Guanacaste", "Cantón": "Santa Cruz", "Distrito": "Tamarindo"},
  "Z": {"Provincia": "Guanacaste", "Cantón": "Nicoya", "Distrito": "Nicoya"}
```

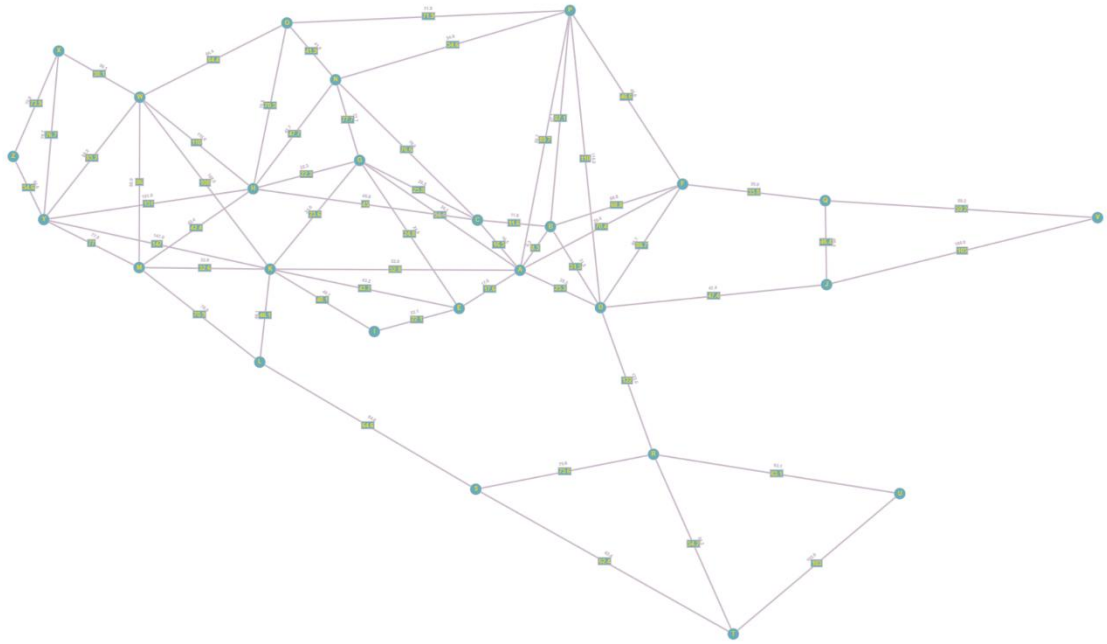

Fuente: *Elaboración propia, captura del JSON en Visual Studio Code*

Mientras que para el grafo y la definición del peso de las asistas:


```
"grafo": {
  "A": [["E", 17.6], ["G", 34.1], ["K", 52.8], ["C", 16.5], ["B", 8.3], ["D", 25.3], ["F", 70.4], ["P", 89.7]],
  "B": [["A", 8.3], ["C", 11.6], ["F", 68.8], ["P", 87.1], ["D", 31.5]],
  "C": [["A", 16.5], ["B", 11.6], ["G", 25.8], ["H", 45.0], ["N", 76.6]],
  "D": [["A", 25.3], ["B", 31.5], ["J", 47.4], ["F", 86.7], ["P", 110.0], ["R", 122.0]],
  "E": [["A", 17.6], ["I", 22.1], ["G", 24.8], ["K", 43.2]],
  "F": [["A", 70.4], ["B", 68.8], ["D", 86.7], ["P", 46.6], ["Q", 35.8]],
  "G": [["A", 34.1], ["C", 25.8], ["E", 24.8], ["H", 22.3], ["K", 23.6], ["N", 77.7]],
  "H": [["C", 45.0], ["G", 22.3], ["N", 47.7], ["O", 70.5], ["M", 42.4], ["W", 110.0], ["Y", 151.0]],
  "I": [["E", 22.1], ["K", 40.1]],
  "J": [["D", 47.4], ["Q", 46.4], ["V", 105.0]],
  "K": [["A", 52.8], ["E", 43.2], ["G", 23.6], ["I", 40.1], ["M", 32.6], ["L", 46.1], ["Y", 147.0], ["W", 108.0]],
  "L": [["K", 46.1], ["S", 64.6], ["M", 70.9]],
  "M": [["H", 42.4], ["K", 32.6], ["L", 70.9], ["Y", 77.0], ["W", 86.0]],
  "N": [["C", 76.6], ["G", 77.7], ["H", 47.7], ["O", 41.9], ["P", 54.6]],
  "O": [["H", 70.5], ["N", 41.9], ["P", 71.9], ["W", 84.4]],
  "P": [["A", 89.7], ["B", 87.1], ["D", 110.0], ["F", 46.6], ["N", 54.6], ["O", 71.9]],
  "Q": [["F", 35.8], ["J", 46.4], ["V", 59.2]],
  "R": [["D", 122.0], ["S", 75.6], ["T", 54.2], ["U", 63.1]],
  "S": [["L", 64.6], ["R", 75.6], ["T", 62.4]],
  "T": [["R", 54.2], ["S", 62.4], ["U", 103.0]],
  "U": [["R", 63.1], ["T", 103.0]],
  "V": [["J", 105.0], ["Q", 59.2]],
  "W": [["H", 110.0], ["K", 108.0], ["M", 86.0], ["O", 84.4], ["X", 50.1], ["Y", 85.2]],
  "X": [["W", 50.1], ["Z", 73.9], ["Y", 76.7]],
  "Y": [["H", 151.0], ["K", 147.0], ["M", 77.0], ["W", 85.2], ["X", 76.7], ["Z", 54.6]],
  "Z": [["X", 73.9], ["Y", 54.6]]
}
```


Fuente: *Elaboración propia, captura del JSON en Visual Studio Code*

Finalmente, exportamos el grafo a Graph Online y también a Neo4j, la primera, para que todos pudiésemos tenerlo a mano de manera práctica sin levantar los contenedores, tratamos de darle la forma mas semejante a donde se ubican las ciudades en el país, y el segundo, en Neo4j mediante el uso de consultas Cypher para construir el grafo dentro de una instancia en Docker que funcione como nuestra base de datos.



Fuente: *Elaboración propia*

Escenarios construidos

Aquí hay un punto importante, y es que diseñamos un código que nos permite rápidamente representar cualquier escenario, sin embargo, para poner en práctica la materia vista en clase y no solo disponer del código sino también del uso de las habilidades adquiridas en el curso diseñamos un pequeño escenario con un caso hipotético posible.

Subimos todo a Google Drive, concretamente en Google Sheets para compartir de forma sencilla cada escenario. Todas las capturas de Google Sheet se podrán ver en el mismo documento.

https://docs.google.com/spreadsheets/d/13_A_8yt9s6fxr3xL7VL8QM4ZhoGbdxxj97ShPb9wAsY/edit?gid=0#gid=0

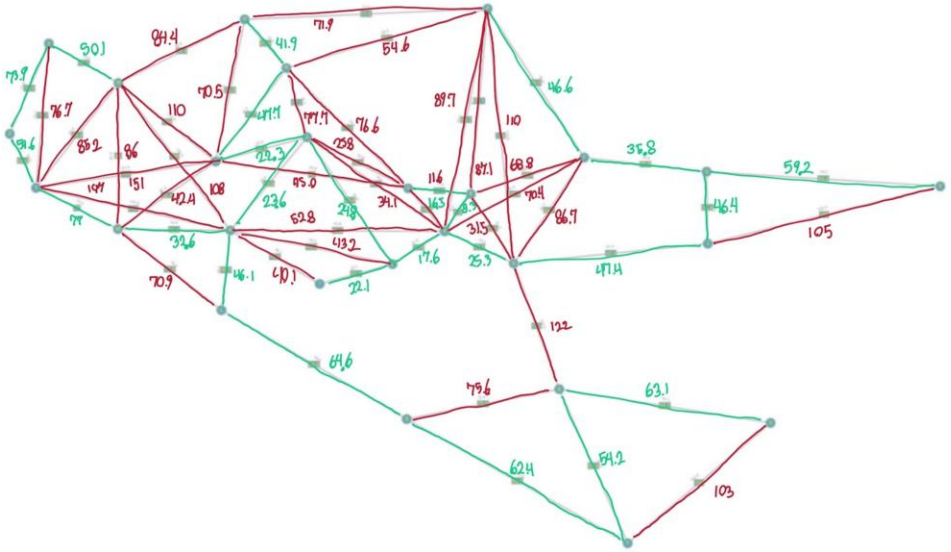
Algoritmo de Dijkstra Aplicado

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26
A	0,A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
B	8.3,A	8.3,A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
C	16.5,A	16.5,A	11.6,B	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
D	25.3,A	25.3,A	25.3,A	25.3,A	25.3,A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
E	17.6,A	17.6,A	17.6,A	17.6,A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
F	70.4,A	70.4,A	70.4,A	70.4,A	70.4,A	70.4,A	70.4,A	70.4,A	70.4,A	70.4,A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
G	34.1,A	34.1,A	34.1,A	34.1,A	34.1,A	34.1,A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
H	?	?	56.6,C	56.6,C	56.6,C	56.4,G	56.4,G	56.4,G	56.4,G	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
I	?	?	?	39.7,E	39.7,E	39.7,E	39.7,E	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
J	?	?	?	?	72.7,D	72.7,D	72.7,D	72.7,D	72.7,D	72.7,D	72.7,D	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
K	52.8,A	52.8,A	52.8,A	52.8,A	52.8,A	52.8,A	52.8,A	52.8,A	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
L	?	?	?	?	?	?	?	98.9,K	98.9,K	98.9,K	98.9,K	98.9,K	98.9,K	98.9,K	98.9,K	*	*	*	*	*	*	*	*	*	*	*
M	?	?	?	?	?	?	?	85.4,K	85.4,K	85.4,K	85.4,K	85.4,K	*	*	*	*	*	*	*	*	*	*	*	*	*	*
N	?	?	88.2,C	88.2,C	88.2,C	88.2,C	88.2,C	88.2,C	88.2,C	88.2,C	88.2,C	88.2,C	88.2,C	*	*	*	*	*	*	*	*	*	*	*	*	*
O	?	?	?	?	?	?	?	?	126.9,H	126.9,H	126.9,H	126.9,H	126.9,H	126.9,H	126.9,H	126.9,H	126.9,H	*	*	*	*	*	*	*	*	*
P	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	89.7,A	*	*	*	*	*	*	*	*	*	*	*	*
Q	?	?	?	?	?	?	?	?	?	106.2,F	106.2,F	106.2,F	106.2,F	106.2,F	106.2,F	106.2,F	*	*	*	*	*	*	*	*	*	*
R	?	?	?	?	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	147.3,D	*	*	*	*	*	*	*	*
S	?	?	?	?	?	?	?	?	?	?	?	?	?	?	163.5,L	163.5,L	163.5,L	163.5,L	163.5,L	163.5,L	163.5,L	*	*	*	*	*
T	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	201.5,R	201.5,R	201.5,R	201.5,R	201.5,R	201.5,R	*	*	*
U	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	210.4,R	210.4,R	210.4,R	210.4,R	210.4,R	210.4,R	210.4,R	*	*
V	?	?	?	?	?	?	?	?	?	?	177.7,J	177.7,J	177.7,J	177.7,J	177.7,J	165.4,Q	165.4,Q	165.4,Q	165.4,Q	165.4,Q	165.4,Q	165.4,Q	*	*	*	*
W	?	?	?	?	?	?	?	160.8,K	160.8,K	160.8,K	160.8,K	160.8,K	160.8,K	160.8,K	160.8,K	160.8,K	160.8,K	160.8,K	160.8,K	*	*	*	*	*	*	*
X	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	210,9,W	210,9,W	210,9,W	210,9,W	210,9,W	210,9,W	210,9,W	*
Y	?	?	?	?	?	?	?	199.8,K	199.8,K	199.8,K	199.8,K	162.4,M	162.4,M	162.4,M	162.4,M	162.4,M	162.4,M	162.4,M	162.4,M	162.4,M	*	*	*	*	*	*
Z	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	217,Y	217,Y	217,Y	217,Y	217,Y	217,Y	217,Y

Fuente: *Elaboración propia*

Algoritmo de Prim:

Las líneas verdes representan la ruta tomada



Nodo	Recorrido	Distancia
A	0	0
B	8.3	8.3
C	19.9	11.6
E	37.5	17.6
I	59.6	22.1
G	84.4	24.8
H	106.7	22.3
K	130.3	23.6
D	155.6	25.3
M	188.2	32.6
L	234.3	46.1
J	281.7	47.4
Q	328.1	46.4
F	363.9	35.8
P	410.5	46.6
N	458.2	47.7
O	500.1	41.9
V	559.3	59.2
S	623.9	64.6
T	686.3	62.4
R	740.5	54.2
U	803.6	63.1
Y	880.6	77
Z	935.2	54.6
X	1009.1	73.9
W	1059.2	50.1

Fuente: *Elaboración propia*

Tabla de Verdad

Tabla de verdad

C1	C2	M1	M2	C1∧C2	¬M1∧¬M2	(C1∧C2) ∧ (¬M1∧¬M2)
0	0	0	0	0	1	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	1	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	**1**	**0**	**0**	**1**	**1**	**1**
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

Fuente: *Elaboración propia*

Ciclos de Mantenimiento

Dia (mod 7)	En Mantenimiento
0	A, I, R, W
1	J, K, P, T
2	C, E, M, X
3	B, H, Q, U
4	O, V, Y
5	F, G, L, Z
6	D, N, S

Formula
d mod 7 = 0

San José (A)	
Dia	Desfase
200	0
Resultado	
FALSE	
Hoy	Dia Futuro
8/8/2025	24/02/2026

Limón (V)	
Dia	Desfase
200	4
Resultado	
TRUE	
Hoy	Dia Futuro
8/8/2025	24/02/2026

Puntarenas (M)	
Dia	Desfase
0	2
Resultado	
FALSE	
Hoy	Dia Futuro
8/8/2025	08/08/2025

Ciudades principales	
A: San José	Mercaderia via Aerea
M: Puntarenas	Mercadería Marítima
V: Limón	Mercadería Marítima

Nota: la meta es que al menos una de estas ciudades siempre esté activa lista para distribuir a las demás sedes

Fuente: *Elaboración propia*

Para los ciclos tenemos la siguiente regla de negocio, no puede haber mas de un centro de distribución principal (Ingreso marítimo o aéreo) en mantenimiento al mismo tiempo, este pequeño sistema en la hoja de cálculo lo que verifica es quien está en mantenimiento (True) y quienes no (False). Nos permite ingresar la cantidad de días, ver la fecha de hoy, qué fecha seria dentro de esa cantidad de días y utilizando colores del semáforo Rojo y Verde nos dice si el lugar está disponible, siempre tienen que haber dos en verde.

Código del Proyecto

Hemos desarrollado un pequeño código para ejecutar cada algoritmo, el código se compone de 4 partes que son.

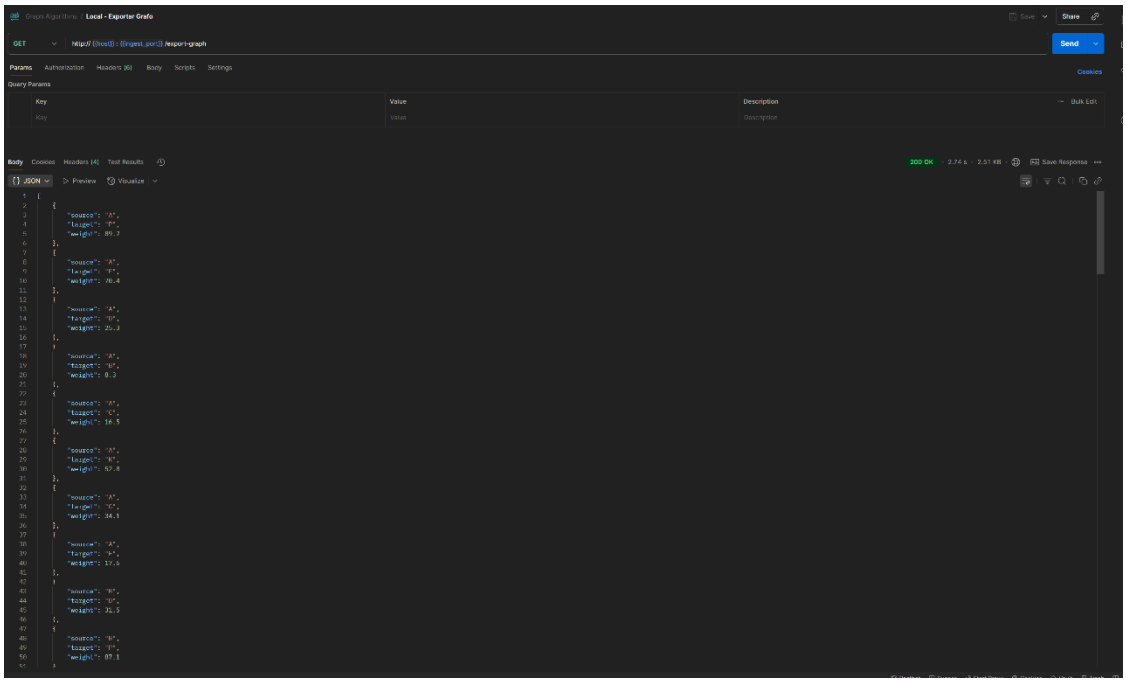
- Fuente de Datos (Instancia levantada en Docker).
- Microservicio para conexión a la base de datos (Ingest-Service).
- Microservicio orquestador del flujo de datos (Routing -Service).
- Repositorio de algoritmos orquestados por Routing-Service (Graph_Analizer).

Los demás componentes son documentación y un logger que nos permitía ir capturando los errores en el código cuando le planteábamos escenarios específicos y así poder entender donde realizar el ajuste. Cada microservicio fue desplegado en Docker y la manera de utilizar el sistema es realizando solicitudes HTTP a localhost en nuestro caso, mediante Postman para facilitar el desarrollo.

Toda la documentación de como levantar el proyecto paso a paso a sido escrita en el Readme del mismo, alojado en un repositorio de GitHub:

https://github.com/KennyWong2024/graph-analysis

Una vez levantado el sistema podemos utilizar el microservicio de ingesta que suministra al orquestador para consultar el grafo y verificar que exista conexión con la base de datos.



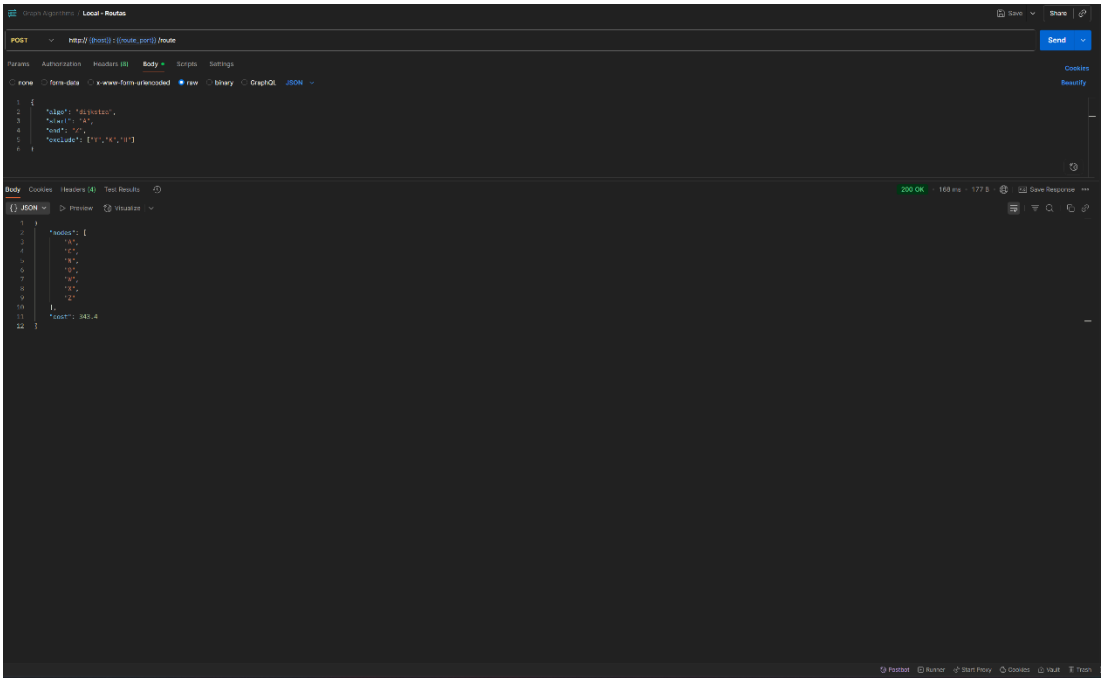
Fuente: *Elaboración propia*

La respuesta esperada debería lucir así por conexión:

```
{
        "source": "A",
        "target": "P",
        "weight": 89.7
    }
```

Donde source es el nodo de partida, target el nodo de destino y weight la distancia en kilómetros que hay entre uno y otro, al tratase de un grafo no dirigido esta distancia es bidireccional.

Por otro lado, podemos consultar al orquestador (Routing-Service) especificando el algoritmo que queremos ejecutar en el Body en formato RAW y el Endpoint nos devolverá la ruta según el algoritmo, para simulación de fallos, podemos añadir una lista de exclusión que emule por ejemplo una ruta cerrada.



Fuente: *Elaboración propia*

CAPÍTULO III: CONCLUSIONES

Recomendaciones

Este proyecto nos permitió explorar todas y cada una de las teorías matemáticas de estructuras discretas pero enfocado en cómo se aplican estos en el día a día, incluso a nivel profesional.

El escenario planteado es solo una pequeña muestra de las aplicaciones en distintos campos, este proyecto nos permitió un espacio de laboratorio en el cual experimentamos y también luchamos por comprender distintas herramientas, pero al final, aprendiendo como sortear estos restos.

Una de las herramientas mas útiles que recomendamos tener a mano es Graph Online, pero utilizarla con criterio, primero desarrollando todo el ejercicio manualmente y luego verificando la integridad de nuestro resultado comparándolo con el de la herramienta.

Finalmente, lo mas importante es aprovechar estos espacios de experimentación para aprender lo máximo posible sin comprometer el entregable, investigar acerca de Cypher, el lenguaje de consulta de Neo4j o incluso explorar la arquitectura de microservicios son algunas de la grandes enseñanzas que nos llevamos de esta experimentación.

Referencias

Para este proyecto no se utilizaron referencias mas allá de los videos complementarios que suministró el profesor para comprender cada algoritmo, el resto fue meramente práctico y utilizando herramientas de las cuales se investigó para entender su funcionamiento mas no ameritan ninguna citación.