GENERADOR DE ITINERARIOS TURÍSTICOS EN BUENOS AIRES

ALUMNA: Narda Rosa.

2025

CARRERA: CIENCIA DE DATOS

PROFESOR: Mauro Cuevas

INDICE

LINK DEL TRABAJO

https://github.com/nardarosa/TP-Final-Narda-Rosa

RESUMEN

Este proyecto presenta el desarrollo de un planificador turístico que genera itinerarios optimizados para recorrer la ciudad de Buenos Aires partiendo desde distintos hoteles y seleccionando un conjunto de atracciones culturales y urbanas. La ciudad se modela como un grafo donde los nodos representan puntos de interés, ya sean hoteles o atracciones, y las aristas indican el tiempo estimado de traslado entre ellos. Para calcular las rutas más eficientes se implementa el algoritmo de Dijkstra, considerando no solo los traslados que se realizan en auto sino también el tiempo estimado de visita de cada atracción. El sistema fue integrado en una aplicación web desarrollada con Flask, que permite al usuario elegir el hotel de inicio, seleccionar las atracciones deseadas y visualizar el itinerario detallado por día y enlaces directos a Google Maps.

INTRODUCCIÓN

Buenos Aires es una de las ciudades con mayor numero de puntos turisticos de interés para los que desean visitarla, estos van desde museos a espacios verdes o monumentos, la mayoria de estos se encuentran distribuidos distintas zonas de la ciudad, alejadas entre si, lo que hace que la organización de un recorrido sea una tarea compleja para la mayoría de los turistas. El que visita Buenos Aires, además, suele tener un tiempo limitado de dias, por lo que elegir la mejor secuencia de lugares para visitar se vuelve muy importante para aprovechar al máximo su estadía.

El problema puede pensarse desde otra perspectiva: ¿cómo recorrer la mayor cantidad de atracciones posibles minimizando los tiempos de traslado? La idea central es representar la ciudad como un grafo, algo muy utilizado para estos tipos de problemas. En este grafo, las atracciones y hoteles son los nodos, mientras que los tiempos de viaje entre ellos constituyen las aristas con pesos.

Sobre este problema se aplica el algoritmo de Dijkstra, un método para obtener caminos mínimos desde un origen hacia todos los demás nodos del grafo, algo aproximado al problema del viajero (directamente relacionado con el dilema de este trabajo). Sin embargo, no solo se busca minimizar distancias: tambien se toma en cuenta un tiempo maximo aproximado de recorrido para cada dia (8hs diarias), dado que conocer nuevos lugares y transportarse desde distintos puntos, hace la persona que lo realiza se canse y no disfrute al maximo su recorrido, ademas también se toma en cuenta el tiempo de permanencia en cada atracción, lo que hace que la elección del próximo punto a visitar sea una combinación entre traslado y visita.

Para dar una solucion y que la vez sea facil de utilizar, se implementó en una aplicación web diseñada con Flask, que permite la interacción directa con el usuario, genera

itinerarios personalizados y ofrece enlaces hacia Google Maps para visualizar los recorridos.

DESCRIPCIÓN

APLICACIÓN DE GRAFOS

La estructura principal es un **grafo no dirigido**, donde:

- **Nodos:** hoteles y atracciones (por ejemplo, *A_Obelisco*, *A_Colon*, *H_Alvear*, etc.).
- **Aristas:** conexiones entre dos puntos con su tiempo estimado de traslado.
- **Peso de arista:** tiempo en minutos.
- **Tiempo de visita:** atributo propio de cada nodo (atracciones).

El grafo se construye a partir de un diccionario llamado GRAFO_TIEMPOS, donde cada clave representa un nodo y como valor tiene un segundo diccionario con sus conexiones más próximas y sus tiempos. A partir de esto, se definen más de 30 nodos entre hoteles y atracciones, y muchas aristas que representan los traslados dentro de la ciudad.

ALGORITMO PRINCIPAL: DIJKSTRA

Para la resolución del problema turístico, el sistema no se limita a un único algoritmo, sino que implementa una solución que combina el algoritmo de Dijkstra para el cálculo de costos de traslado y una estrategia voraz para la toma de decisiones secuenciales bajo restricciones temporales.

ALGORITMO DE DIJKSTRA OPTIMIZADO

La función de este algoritmo dentro del sistema es actuar como un "radar" de distancias. Dado un nodo de origen (la ubicación actual del turista), Dijkstra calcula el tiempo mínimo necesario para llegar a todos los demás nodos del grafo (atracciones y hoteles).

Para garantizar la eficiencia, se utiliza una cola de prioridad (mediante la librería heapq).

- **Eficiencia:** A diferencia de una búsqueda lineal en una lista desordenada, la cola de prioridad mantiene los nodos ordenados internamente por distancia acumulada.
- **Funcionamiento:** Esto permite extraer siempre el nodo más cercano accesible en tiempo constante, asegurando que el algoritmo expanda siempre el camino más prometedor primero.

ESTRATEGIA VORAZ

Una vez que Dijkstra proporciona las distancias hacia todas las atracciones, el sistema debe decidir cuál visitar a continuación. Aquí se aplica una estrategia voraz. El algoritmo selecciona la atracción pendiente más cercana con la expectativa de construir un itinerario global eficiente.

FLUJO DE EJECUCIÓN

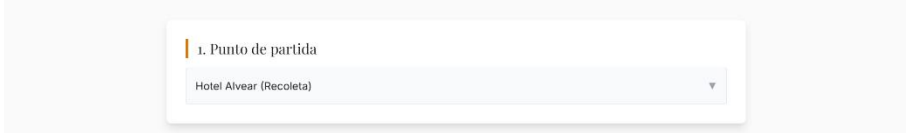
El ciclo de generación del itinerario para un día cualquiera sigue el siguiente procedimiento lógico:

1. **Inicio:** El día comienza con el turista ubicado en el hotel seleccionado.
2. **Cálculo de Distancias:** Se ejecuta el algoritmo de Dijkstra desde el nodo actual para actualizar los tiempos de viaje hacia todos los puntos de la red.
3. **Filtrado de Pendientes:** Se genera una sub-lista que contiene únicamente las atracciones que aún no han sido visitadas.
4. **Selección:** De la lista de pendientes, se identifica la atracción que posee el menor tiempo de traslado desde la ubicación actual.
5. **Cálculo del Costo Real:** Se determina el costo total que implicaría realizar esta visita, sumando dos variables:
    a. *Costo de la Arista:* Tiempo de viaje (obtenido por Dijkstra).
    b. *Peso del Nodo:* Tiempo estimado de visita turística (propio de cada atracción).
6. **Validación de Restricción Temporal:** Se verifica si la suma del tiempo acumulado del día + costo total del nuevo tramo es menor o igual al límite establecido de 480 minutos (8 horas).
    a. **Caso Afirmativo:** Se confirma la visita, se actualiza la ubicación actual al nuevo nodo, se suma el tiempo y se repite el ciclo desde el paso 2.
    b. **Caso Negativo:** Si la visita excede el tiempo límite, se descarta la atracción para ese día. El algoritmo fuerza el cierre de la jornada calculando el regreso al hotel y finaliza el itinerario del día corriente.

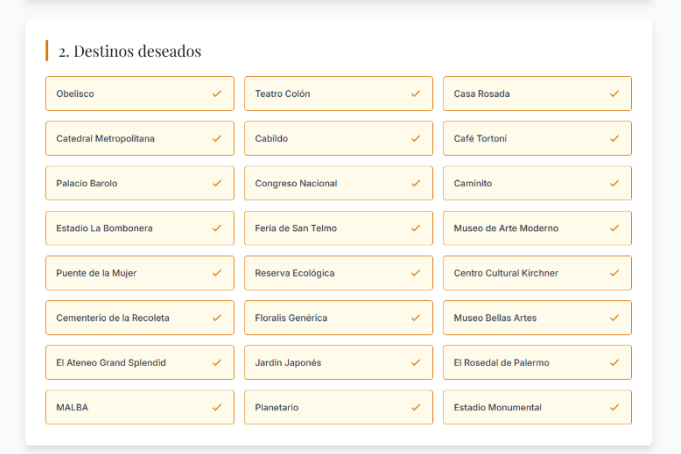
INTERFAZ WEB CON FLASK

El funcionamiento del proyecto se integra en una aplicación web:

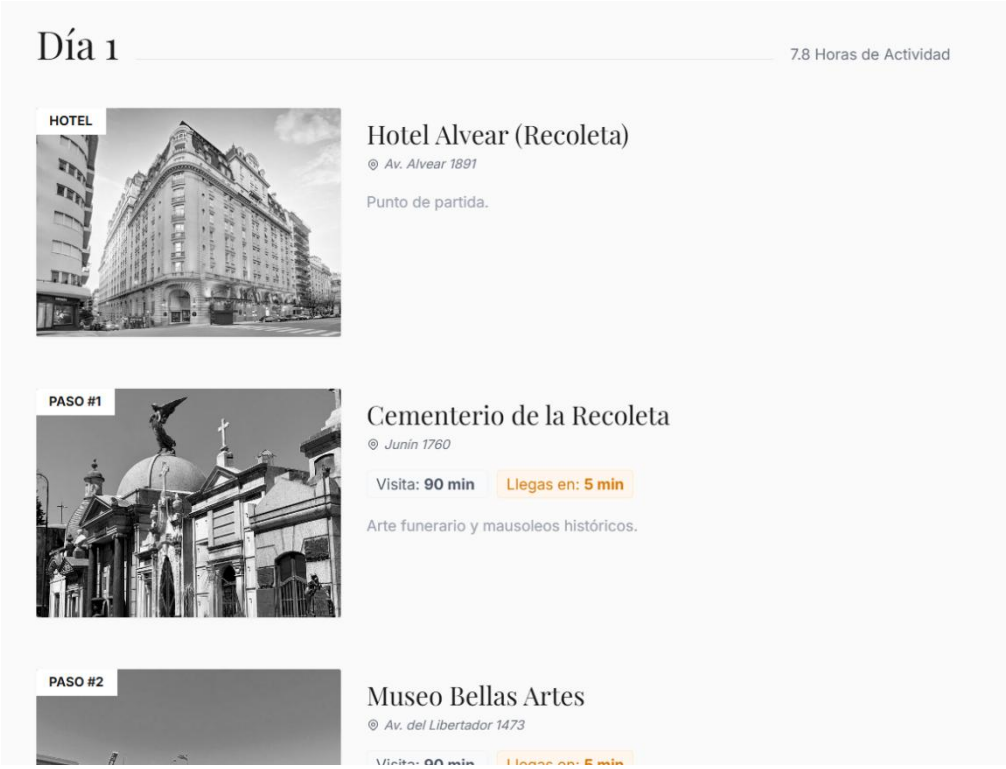
- Página principal: con un formulario donde el usuario elige:
    - hotel de inicio



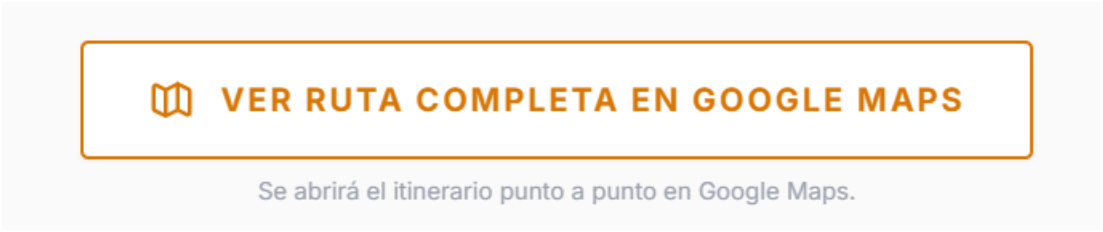
- atracciones a visitar (checkboxes)



- Backend: recibe los datos y ejecuta el algoritmo.
- Plantilla HTML que muestra:
  - itinerario día por día
  - imágenes de cada lugar
  - tiempos de visita y traslado



  - botón “Ver ruta en Google Maps”, generado automáticamente.



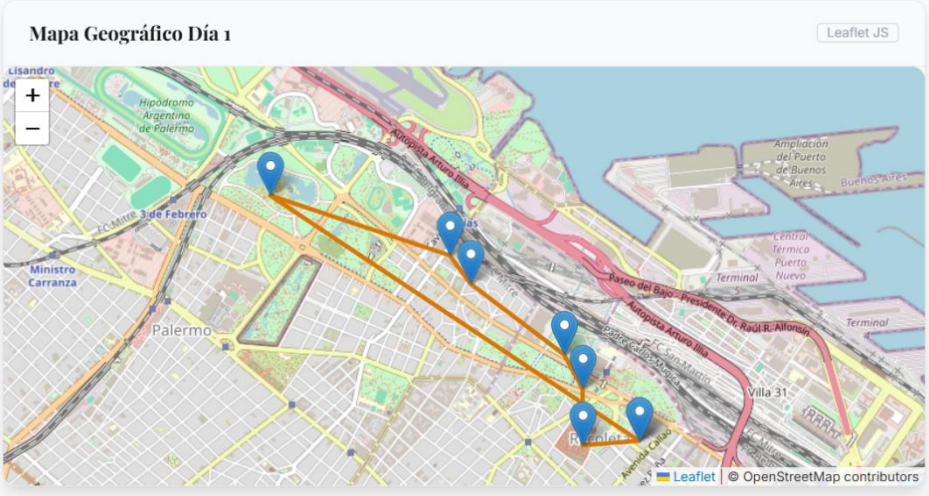
- demostración de la minimizacion, (funcionamiento de Dijkstra).

Demostración de Minimización

En cada paso, el algoritmo evaluó los pesos w(u, v) de los vecinos pendientes y seleccionó el mínimo.

PASO	DESDE (U)	DECISIÓN (MIN)	OTRAS OPCIONES EVALUADAS
#1	Hotel Alvear (Recoleta)	→ Cementerio de la Recoleta (w=5)	• Museo Bellas Artes (w=10) ✕ • Teatro Colón (w=15) ✕ • El Ateneo Grand Splendid (w=15) ✕
#2	Cementerio de la Recoleta	→ Museo Bellas Artes (w=5)	• El Ateneo Grand Splendid (w=10) ✕ • Floralis Genérica (w=10) ✕ • Teatro Colón (w=15) ✕
#3	Museo Bellas Artes	→ Floralis Genérica (w=5)	• El Ateneo Grand Splendid (w=15) ✕ • Jardín Japonés (w=15) ✕ • MALBA (w=15) ✕
#4	Floralis Genérica	→ MALBA (w=10)	• Jardín Japonés (w=15) ✕ • El Ateneo Grand Splendid (w=20) ✕ • Teatro Colón (w=25) ✕
#5	MALBA	→ Jardín Japonés (w=5)	• El Rosedal de Palermo (w=15) ✕ • Planetario (w=15) ✕ • El Ateneo Grand Splendid (w=30) ✕
#6	Jardín Japonés	→ El Rosedal de Palermo (w=10)	• Planetario (w=10) ✕ • Estadio Monumental (w=25) ✕ • El Ateneo Grand Splendid (w=30) ✕

- Mapa interactivo hecho en Leaflet, donde se puede visualizar los puntos a visitar en el dia en modo de grafo.



La base de la solucion de proyecto esta hecho en **Python** y la interfaz está desarrollada con **HTML, Jinja2 y TailwindCSS.**

RESULTADOS

Los resultados se obtuvieron ejecutando el algoritmo desde los 5 distintos hoteles de la ciudad disponibles y seleccionando todas las atracciones disponibles.

RESULTADOS EN PÁGINA WEB

Las tablas generadas en la web muestran, para cada día:

- nombre de la atracción
- tiempo de viaje desde el punto anterior
- tiempo estimado que lleva recorrer la atracción

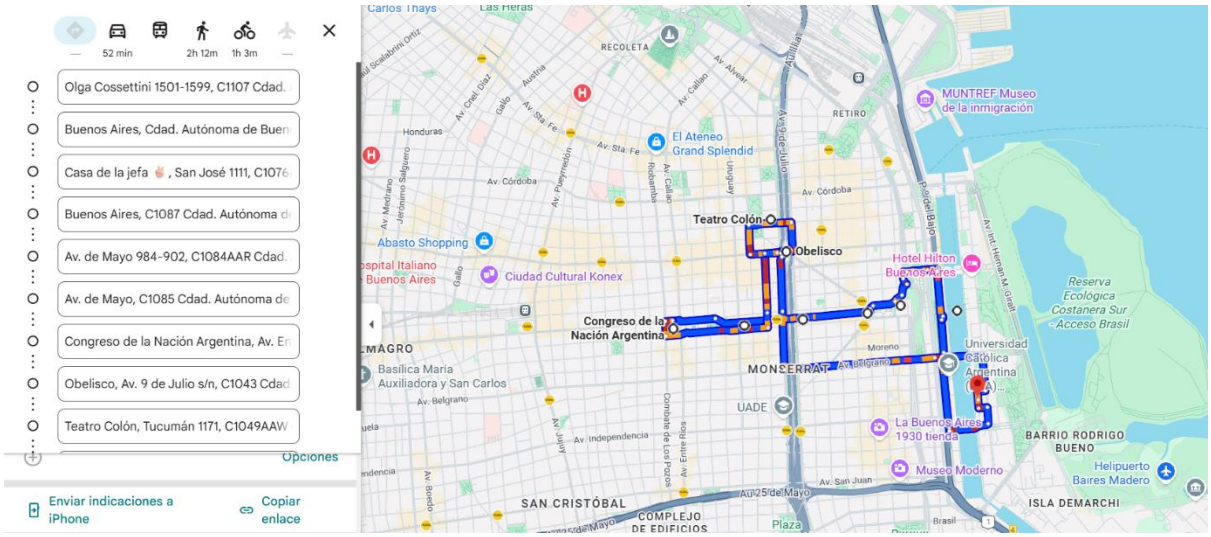
También incluye:

- total de horas utilizadas por día
- total de atracciones visitadas
- tiempo total del recorrido

(Todo visto anteriormente)

MAPAS DEL RECORRIDO

Cada día se acompaña con un enlace a Google Maps generado automáticamente mediante las coordenadas almacenadas en el diccionario COORDENADAS.
Esto permite visualizar el recorrido real punto por punto en un mapa totalmente navegable, útil para dispositivos móviles.



DEMOSTRACIÓN DE MINIMIZACIÓN

Para mostar la logica el funcionamiento de Dijkstra se agregó un cuadro, donde de ve el registro de cada decisión que toma el algoritmo y lo muestra en una tabla desplegable. Ahi se puede ver paso a paso qué opciones evaluó el algoritmo, sus pesos y cuál eligió por ser la mínima, dejando en evidencia cómo funciona la estrategia voraz.

MAPA INTERACTIVO

Para mostar la logica el funcionamiento de Dijkstra se agregó un cuadro, donde de ve el registro de cada decisión que toma el algoritmo y lo muestra en una tabla desplegable. Ahi se puede ver paso a paso qué opciones evaluó el algoritmo, sus pesos y cuál eligió por ser la mínima, dejando en evidencia cómo funciona la estrategia voraz.

RESULTADOS EN CONSOLA

Para mostrar los resultados se eligio un hotel en modo de ejemplo (Hotel Hilton), y se ve el itinerario de un solo dia para que el ejemplo no sea muy extenso.

```
============================================================
RESUMEN DEL ITINERARIO
============================================================
Hotel de Inicio:       Hotel Hilton (Puerto Madero)
Total Atracciones:     24
Tiempo Total Global:   1862 minutos
============================================================

 >>> DÍA 1 <<<
 Tiempo del día: 477 minutos
 Link del Mapa: https://www.google.com/maps/dir/-34.6103,-58.364/-34.6084,-58.365/-
34.608,-58.3705/-34.6076,-58.3733/-34.6086,-58.3737/-34.6091,-58.38/-34.6096,-58.3857/-
34.6098,-58.3926/-34.6037,-58.3816/-34.6011,-58.3831/-34.6103,-58.364
------------------------------------------------------------
PASO   LUGAR                                VIAJE      VISITA
------------------------------------------------------------
INI    Hotel Hilton (Puerto Madero)         0 min      0 min
1      Puente de la Mujer                   5 min      20 min
2      Casa Rosada                          10 min     60 min
3      Catedral Metropolitana               5 min      30 min
4      Cabildo                              2 min      45 min
5      Café Tortoni                         5 min      45 min
6      Palacio Barolo                       5 min      90 min
7      Congreso Nacional                    5 min      30 min
8      Obelisco                             15 min     15 min
9      Teatro Colón                         5 min      60 min
FIN    Hotel Hilton (Puerto Madero) (Reg    25 min     0 min
.  .  .
```

CONCLUSIÓN

Aunque el problema se parece al problema del viajero, en este proyecto se eligió usar el algoritmo de Dijkstra porque es rápido, eficiente y permite armar recorridos que tienen sentido en la práctica, gracias a eso, el sistema puede calcular rutas de forma inmediata y ofrecer itinerarios que realmente sirven para un turista.

También se sumaron dos elementos importantes para hacerlo más realista: el tiempo de visita de cada atracción y un límite de ocho horas por día. Esto permite que los recorridos generados sean parecidos a lo que una persona podría hacer en un día normal de paseo, sin tiempos imposibles o recorridos exagerados.

Además, integrar el algoritmo dentro de una aplicación web y sumar enlaces a Google Maps hace que todo el proceso sea fácil de usar, visual y accesible para cualquiera, un mapa interactivo para ver el recorrido y como realemnte se evaluo los lugares para asi ofrecer el mejor itinerario posible. En conjunto, el proyecto muestra que es posible crear una herramienta simple y práctica para planificar recorridos por Buenos Aires usando ideas básicas de grafos y programación.