Universidad de Nacional de San Agustín de Arequipa
Facultad de Ingeniería Producción y Servicios
Escuela Profesional Ciencia de la Computación

Informe Final de Análisis Exploratorio de datos del conjunto de datos GTFS estáticos del Sistema De Transporte Público de San Francisco(SFMTA)

Tópicos en Ciencia de Datos

Integrantes: Kemely Francis Castillo Caccire
Docente: Ana Maria Cuadros Valdivia

Fecha de entrega: 5 de junio del 2025

Arequipa, Perú

1. Hipótesis iniciales

1.1. Motivación

Aunque el formato GTFS es muy usado para compartir datos de transporte público, está dividido en varios archivos (`routes.txt`, `stops.txt`, `trips.txt`, `stop_times.txt`, `calendar.txt`, etc.) y eso lo hace difícil de entender y procesar rápidamente. Como señalan Wu et al. [3], manejar estos datos de GTFS puede resultar complicado debido a la cantidad de información y a la forma en que se relacionan entre sí. Por otro lado, herramientas como G2Viz [2] demuestran que, si logramos unir y limpiar esos datos, podemos crear gráficos y mapas muy útiles (por ejemplo, mapas de calor de paradas o curvas de frecuencia) que ayudan a planificadores y usuarios a ver claramente cómo funciona el sistema. Partiendo de esa idea—que GTFS es valioso pero complejo—planteamos hipótesis para entender mejor la red de transporte de SFMTA.

1.2. Hipótesis

Hipótesis 1

¿Las rutas que tienen más paradas tienen menos viajes diarios que las rutas con menos paradas?

Si una ruta pasa por muchas paradas, cada recorrido dura más tiempo. Por eso es probable que pongan menos viajes en esas rutas largas para no atrasarse y mantener un servicio ordenado.

Hipótesis 2

¿Los días laborales tienen dos picos de servicio (7–9 AM y 4–7 PM), mientras que los fines de semana hay un servicio más constante y menos viajes totales?

Entre semana, la gente va a trabajar o a estudiar, así que hay mucha demanda a primera hora y en la tarde. En fin de semana la actividad baja y los viajes se reparten más parejo, sin saltos tan fuertes.

Hipótesis 3

¿El centro de San Francisco tiene muchas más paradas por área que las zonas alejadas, mostrando que algunos barrios tienen menos cobertura?

En el centro hay oficinas, tiendas y transporte pesado, por eso concentran más paradas. En barrios alejados (como Sunset o Twin Peaks) las paradas están más separadas, lo que dificulta que la gente tenga un autobús cerca.

1.3. Plan de análisis

1.3.1. Hipótesis 1

- **Objetivo:** Ver si las rutas con más paradas tienen menos viajes diarios que las rutas con menos paradas.
- **Pasos:**
  - Limpiar y unir `trips.txt` con `routes.txt`.
  - Contar cuántos viajes diarios (`num_trips`) tiene cada `route_id`.
  - Con `stop_times.txt`, contar cuántas paradas únicas (`num_stops`) cubre cada ruta.
  - Crear un scatterplot de `num_stops` vs. `num_trips`.
  - Calcular la correlación de Pearson para ver si existe relación negativa.

1.3.2. Hipótesis 2

- **Objetivo:** Ver si los días laborales tienen picos de servicio (7–9 AM y 4–7 PM) y si los fines de semana son más uniformes.
- **Pasos:**
  - Filtrar `stop_times.txt` para viajes activos en días laborables y en fines de semana (usando `calendar.txt` y `calendar_dates.txt`).
  - Convertir `arrival_time` a segundos y agrupar por hora para contar llegadas (`num_arrivals`) en cada hora.
  - Graficar `num_arrivals` por hora para días laborales y para fines de semana.
  - Calcular headways: ordenar llegadas por `stop_id` y hora, medir tiempo entre llegadas sucesivas, agrupar por hora.
  - Dibujar un boxplot de headways por hora para comparar horas punta y horas de baja demanda.

1.3.3. Hipótesis 3

- **Objetivo:** Ver si el centro de San Francisco tiene más paradas por área que las zonas alejadas.
- **Pasos:**
  - Limpiar `stops.txt`: eliminar paradas fuera del rango de latitud/longitud de San Francisco.
  - Redondear `stop_lat` y `stop_lon` a dos decimales para crear celdas de 0.01°.
  - Contar cuántas paradas hay en cada celda geográfica (`num_stops` por celda).
  - Etiquetar celdas como “centro” o “periferia” según coordenadas.
  - Generar un heatmap (mapa de calor) de densidad de paradas usando las celdas.
  - Calcular la densidad promedio (paradas/km²) en zona centro vs. zona periferia.
  - Comparar ambos valores para confirmar la brecha de cobertura.

2. Fuente de datos

2.1. Fuente

Origen y recolección El feed GTFS de SFMTA (`muni_gtfs-current.zip`) se descargó el 1 de junio de 2025 desde https://www.sfmta.com/reports/gtfs-transit-data. Lo publica mensualmente la San Francisco Municipal Transportation Agency (SFMTA), usando datos de:

- Dispositivos GPS instalados en cada vehículo para obtener coordenadas en tiempo real.
- Planillas internas de rutas y horarios, exportadas al formato GTFS.
- Sistemas de control de operaciones que consolidan estos datos en un "snapshot" estático cada mes.

Para comprender el estándar GTFS se utilizó la especificación oficial disponible en https://gtfs.org/reference/static/ [1].

Área de conocimiento y problema a resolver Este conjunto pertenece al área de Transporte Público Urbano y Planificación de Movilidad. Integra conceptos de:

- **Sistemas de Información Geográfica (GIS):** uso de coordenadas `stop_lat` y `stop_lon` para visualizar y analizar la distribución espacial de paradas [4].
- **Operación de Flotas y Programación de Horarios:** cálculo de headways según el *Transit Capacity and Quality of Service Manual* (TCQSM) [5], donde se definen métricas para intervalos de servicio.
- **Análisis de Datos Masivos:** manejo de grandes archivos CSV (p. ej., `stop_times.txt` 3 GB) usando `pandas` [6] y visualizaciones con `Matplotlib` [7].

2.2. Descripción

A nivel de atributos

Archivo	Codificación	Tamaño (MB)	Granularidad
agency.txt	ascii	0.000000	1 filas x 8 columnas
stops.txt	ascii	0.272000	3278 filas x 8 columnas
routes.txt	ascii	0.006000	69 filas x 10 columnas
trips.txt	ascii	1.885000	35017 filas x 9 columnas
stop_times.txt	ascii	54.836000	1313881 filas x 9 columnas
calendar.txt	ascii	0.000000	3 filas x 10 columnas
calendar_dates.txt	ascii	0.000000	26 filas x 3 columnas
fare_attributes.txt	ascii	0.000000	2 filas x 7 columnas
fare_rules.txt	ascii	0.000000	69 filas x 2 columnas
shapes.txt	ascii	1.798000	46249 filas x 5 columnas
directions.txt	ascii	0.002000	138 filas x 3 columnas
timepoints.txt	ascii	4.527000	249904 filas x 2 columnas

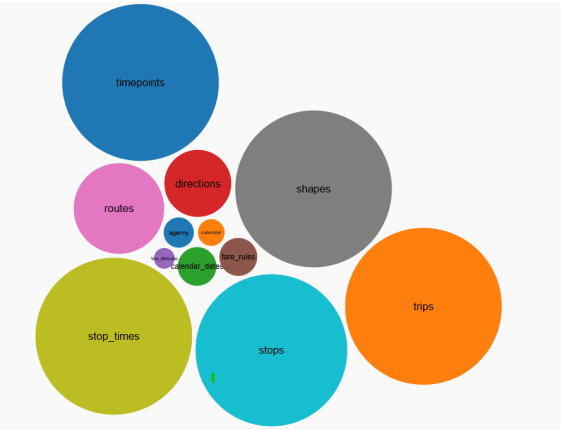


Figura 1: Visualización de archivos GTFS

- El conjunto de datos GTFS tiene 12 archivos con diferentes tamaños, codificación y granularidad, reflejando la complejidad del transporte público.
- Los archivos `stop_times.txt` y `trips.txt` son los más grandes y con mayor granularidad, mostrando detalles finos de horarios y viajes.
- Archivos como `calendar.txt` y `fare_rules.txt` son pequeños y cumplen funciones auxiliares.
- La mayoría de los archivos usan codificación ASCII, facilitando su lectura y manipulación.
- La granularidad varía desde pocos registros con muchas columnas hasta muchos registros con pocas columnas.

Se identificaron campos específicos con presencia de datos faltantes, los cuales se resumen a continuación:

Tabla	Estado de Calidad	Observación
agency	Completa (sin nulos)	Datos listos para análisis
stops	Todas filas con nulos, requiere limpieza	Clave para ubicación, limpieza crítica
routes	Mayoría con nulos, revisar columnas	Nulos en algunas filas, evaluar columnas
trips	Completa (sin nulos)	Datos completos, buen nivel de calidad
stop_times	Todas filas con nulos, requiere limpieza	Tabla muy grande, limpieza imprescindible
calendar	Completa (sin nulos)	Datos completos, pocas filas
calendar_dates	Completa (sin nulos)	Datos completos, pocas filas
fare_attributes	Mayoría con nulos, revisar columnas	Pocas filas completas, evaluar columnas
fare_rules	Completa (sin nulos)	Datos completos, buen nivel de calidad
shapes	Completa (sin nulos)	Datos completos, buen nivel de calidad
directions	Completa (sin nulos)	Datos completos, buen nivel de calidad
timepoints	Completa (sin nulos)	Datos completos, buen nivel de calidad

A nivel de registros

Analizamos las tablas en un :colab y sus contenidos en orden a su importancia.

Requeridos

- **agency**

	agency_id	agency_name	agency_url	agency_timezone	agency_lang	agency_phone	agency_fare_url	agency_email
0	SFMTA	San Francisco Municipal Transportation Agency	http://www.sfmta.com	America/Los_Angeles	en	311	https://SFMTA.com/Fares	munifeedback@sfmta.com

Figura 2: 3 primeros registros de la tabla agency

- **Una ciudad puede tener una o varias agencias.**
  El archivo puede listar múltiples agencias. Por ejemplo, San Francisco usa solo SFMTA, pero ciudades como Arequipa en nuestro caso pueden tener varias hasta 10 [**?**].
- **La zona horaria es clave.**
  El campo `agency_timezone` asegura que los horarios se muestren correctamente en apps. Si está mal configurado, los horarios se desajustan [**?**].
- **El número de teléfono no siempre es internacional.**
  Por ejemplo, San Francisco usa "311", un número local. No todos los feeds usan formato internacional, lo cual puede generar confusión [**?**].
- **El idioma afecta cómo se ve todo.**
  El campo `agency_lang` define el idioma predeterminado del feed, lo que influye en cómo se muestra la información en las apps [**?**].
- **No todos los campos son obligatorios.**
  Campos como `agency_email` o `agency_fare_url` son opcionales, pero útiles para el contacto o información adicional [**?**].

- **stops**

```
--- stops.txt: primeros 3 registros ---
```

	stop_id	stop_code	stop_name	stop_desc	stop_lat	stop_lon	zone_id	stop_url
0	390	10390	19th Avenue & Holloway St		37.721190	-122.475153		https://SFMTA.com/10390
1	913	10913	Dublin St & La Grande Ave		37.719192	-122.425802		https://SFMTA.com/10913
2	3016	13016	3rd St & 4th St		37.772618	-122.389786		https://SFMTA.com/13016
3	3018	13018	Bacon St & San Bruno Ave		37.727859	-122.402994		https://SFMTA.com/13018
4	3019	13019	Bacon St & San Bruno Ave		37.727645	-122.403269		https://SFMTA.com/13019
5	3020	13020	Bacon St & Somerset St		37.726670	-122.407460		https://SFMTA.com/13020
6	3021	13021	Bacon St & Somerset St		37.726540	-122.407660		https://SFMTA.com/13021
7	3023	13023	Baker Beach Parking Lot SW		37.790249	-122.482263		https://SFMTA.com/13023
8	3024	13024	Baker St & Greenwich St		37.797606	-122.445693		https://SFMTA.com/13024
9	3031	13031	Bay St & Midway St		37.806094	-122.409161		https://SFMTA.com/13031

Figura 3: 3 primeros registros de la tabla stops

- Un mismo nombre de parada puede tener varios `stop_id` diferentes. Por ejemplo, *Bacon St & San Bruno Ave* aparece con varios `stop_id` distintos (3018 y 3019). Esto ocurre porque cada lado de la calle o sentido puede tener una parada distinta, aunque el nombre sea igual.
- `stop_code` puede ser distinto de `stop_id` y funciona como código para usuarios. `stop_code` suele ser un número más corto o amigable que los pasajeros pueden ver en señales o apps, mientras que `stop_id` es un identificador único para el sistema.
- Algunas paradas no tienen descripción (`stop_desc`) pero sí URL con más información. Aunque el campo `stop_desc` esté vacío, el campo `stop_url` ofrece un enlace donde se puede obtener más datos o mapas, lo que ayuda a usuarios que quieren detalles extras.

- Las coordenadas geográficas (`stop_lat` y `stop_lon`) no solo marcan la ubicación, sino que permiten calcular distancias y rutas automáticamente. Esto facilita la generación automática de mapas, cálculo de distancias a pie o la navegación GPS para apps.
- La ausencia de `zone_id` indica que el sistema no usa zonas tarifarias en algunas paradas. Si bien `zone_id` sirve para definir tarifas basadas en zonas geográficas, en este caso muchas paradas no tienen zona, probablemente porque la tarifa es fija o la ciudad no usa zonas para precios.
- Las paradas pueden ser puntos muy cercanos, incluso a pocos metros, para diferentes direcciones o rutas. Esto pasa mucho en cruces grandes, donde cada dirección de viaje tiene su parada propia, a veces justo en lados opuestos de la calle.
- El sistema permite representar desde paradas simples en la calle hasta lugares complejos como estacionamientos o terminales. Por ejemplo, *Baker Beach Parking Lot SW* no es una parada normal sino un área específica para transporte, lo que muestra la flexibilidad del formato para distintos tipos de puntos de parada.

- **routes**

--- routes.txt: primeros 3 registros ---

	route_id	agency_id	route_short_name	route_long_name	route_url	route_desc	route_type	route_color	route_text_color	route_sort_order
0	714	SFMTA	714	BART EARLY BIRD	http://www.sfmta.com/714	Weekdays 4am-5am	3	005B95	FFFFFF	NaN
1	8	SFMTA	8	BAYSHORE	http://www.sfmta.com/8	5am-12 midnight daily	3	005B95	FFFFFF	NaN
2	8AX	SFMTA	8AX	BAYSHORE A EXPRESS	http://www.sfmta.com/8AX	Weekdays 6:30-9:30am (northbound) 3:30-6:50pm (southbound)	3	005B95	FFFFFF	NaN
3	8BX	SFMTA	8BX	BAYSHORE B EXPRESS	http://www.sfmta.com/8BX	Weekdays 6:30-9am (northbound) 3:30-6:30pm (southbound)	3	005B95	FFFFFF	NaN
4	9	SFMTA	9	SAN BRUNO	http://www.sfmta.com/9	5am-12 midnight daily	3	005B95	FFFFFF	NaN
5	90	SFMTA	90	SAN BRUNO OWL	http://www.sfmta.com/90	12 midnight-5am daily	3	666666	FFFFFF	NaN
6	91	SFMTA	91	3RD-19TH AVE OWL	http://www.sfmta.com/91	12 midnight-5am daily	3	666666	FFFFFF	NaN
7	9R	SFMTA	9R	SAN BRUNO RAPID	http://www.sfmta.com/9R	Weekdays 7am-6pm	3	BF2B45	FFFFFF	NaN
8	CA	SFMTA	CA	CALIFORNIA STREET CABLE CAR	http://www.sfmta.com/CA	7 am-8:30 pm daily	5	B49A36	000000	NaN
9	F	SFMTA	F	MARKET & WHARVES	http://www.sfmta.com/F	7am-10pm daily	0	B49A36	000000	NaN

Figura 4: 3 primeros registros de la tabla routes

- Los IDs como 8, 8AX y 8BX son rutas relacionadas pero diferentes. Aunque comparten el número base (8), representan variaciones con diferentes horarios o paradas. El sufijo (AX, BX, R) indica rutas expresas o rápidas. Esto no siempre está estandarizado, así que las aplicaciones deben interpretar bien estos nombres.
- Hay rutas que sólo operan en horarios extremos. Por ejemplo, 714 BART EARLY BIRD opera sólo de 4 am a 5 am, y 90 SAN BRUNO OWL de 12 am a 5 am. Estas rutas "temprano" o "nocturnas" existen para cubrir la ciudad cuando el sistema principal (como BART) está cerrado.
- El mismo número puede aparecer en una versión diurna y otra nocturna. La ruta 9 es "SAN BRUNO", y 90 es "SAN BRUNO OWL", su equivalente nocturno. No siempre es obvio para el usuario casual que están relacionadas.
- Los colores ayudan a distinguir tipos de servicio. Rutas normales usan azul (005B95), rápidas usan rojo (BF2B45), y nocturnas usan gris (666666). Aunque el GTFS no impone reglas sobre colores, algunas agencias lo usan para comunicar visualmente el tipo de servicio.
- El tipo de transporte es más diverso de lo que parece. El campo `route_type` = 3 indica autobuses, pero 5 es para tranvías (como el Cable Car), y 0 indica tranvía histórico

(como la ruta F de Market & Wharves). Esta clasificación permite representar muchos modos en una sola estructura.

- Algunas rutas tienen nombres largos y promocionales. Ejemplo: CALIFORNIA STREET CABLE CAR no es un bus regular, sino una atracción turística reconocida, y su nombre lo deja claro. Esto es útil para turistas pero puede confundir al integrarse con rutas regulares.
- El campo `route_sort_order` está vacío, pero podría usarse para ordenar visualmente. Si se completa, permite que apps muestren las rutas en orden lógico, como primero locales, luego expresas, luego nocturnas. Que esté en blanco significa que el orden depende de cada app o queda al azar.

- **trips**

--- trips.txt: primeros 3 registros ---

	route_id	service_id	trip_id	trip_headsign	direction_id	block_id	shape_id	wheelchair_accessible	bikes_allowed
0	1	M12	11708585_M12	Geary + 33rd Avenue	0	109_M12	102	1	1
1	1	M12	11708586_M12	Presidio Avenue	0	105_M12	101	1	1
2	1	M12	11708587_M12	Presidio Avenue	0	101_M12	101	1	1
3	1	M12	11708588_M12	Presidio Avenue	0	104_M12	101	1	1
4	1	M12	11708589_M12	Geary + 33rd Avenue	0	102_M12	102	1	1
5	1	M12	11708590_M12	Geary + 33rd Avenue	0	107_M12	102	1	1
6	1	M12	11708591_M12	Geary + 33rd Avenue	0	110_M12	102	1	1
7	1	M12	11708592_M12	Geary + 33rd Avenue	0	103_M12	102	1	1
8	1	M12	11708593_M12	Geary + 33rd Avenue	0	101_M12	102	1	1
9	1	M12	11708594_M12	Geary + 33rd Avenue	0	104_M12	102	1	1

Figura 5: 3 primeros registros de la tabla trips

- Un mismo `route_id` puede tener muchos `trip_id` distintos. Esto significa que una ruta puede tener varios viajes diferentes durante el día, cada uno con su propio identificador para organizar horarios y recorridos.
- El `service_id` define cuándo opera un viaje. Por ejemplo, puede indicar si el viaje es solo en días laborables, fines de semana o días festivos, permitiendo controlar el calendario del servicio.
- El `trip_headsign` es el texto que aparece en el bus o app para mostrar el destino. Aunque la ruta sea la misma, diferentes viajes pueden tener destinos distintos para informar a los pasajeros.
- El `direction_id` usualmente es 0 o 1 y marca ida o vuelta. Esto ayuda a diferenciar viajes en ambas direcciones dentro de una misma ruta.
- El `block_id` agrupa viajes que se hacen con el mismo bus consecutivamente. Así, un vehículo puede cubrir varios viajes sin regresar a la base, optimizando recursos.
- El `shape_id` indica la ruta exacta o forma que sigue el viaje en el mapa. Esto permite representar con precisión el recorrido en sistemas de mapas o apps.

- El campo `wheelchair_accessible` muestra si el viaje es accesible para personas con discapacidad motriz. Valor 1 indica accesible, 0 no accesible o desconocido.
- El campo `bikes_allowed` indica si está permitido llevar bicicletas en ese viaje. Esto es útil para pasajeros que usan bicicleta y el transporte público.
- Aunque los viajes compartan ruta y servicio, pueden tener destinos, accesibilidad o recorridos diferentes. Esto refleja la flexibilidad del sistema para manejar variantes dentro de una misma ruta.

- **stop_times**
  Cada registro en stop_times representa un paso de un viaje (`trip`) por una parada específica, con hora de llegada y salida, y el orden en la secuencia de paradas.

--- stop_times.txt: primeros 3 registros ---

	trip_id	arrival_time	departure_time	stop_id	stop_sequence	stop_headsign	pickup_type	drop_off_type	shape_dist_traveled
0	11708204_M21	04:32:00	04:32:00	3892	1	NaN	NaN	NaN	NaN
1	11708204_M21	04:32:41	04:32:41	3875	2	NaN	NaN	NaN	NaN
2	11708204_M21	04:33:28	04:33:28	3896	3	NaN	NaN	NaN	NaN
3	11708204_M21	04:34:12	04:34:12	3852	4	NaN	NaN	NaN	NaN
4	11708204_M21	04:34:45	04:34:45	3845	5	NaN	NaN	NaN	NaN
5	11708204_M21	04:35:25	04:35:25	3822	6	NaN	NaN	NaN	NaN
6	11708204_M21	04:36:00	04:36:00	3824	7	NaN	NaN	NaN	NaN
7	11708204_M21	04:36:36	04:36:36	7160	8	NaN	NaN	NaN	NaN
8	11708204_M21	04:37:15	04:37:15	3828	9	NaN	NaN	NaN	NaN
9	11708204_M21	04:37:43	04:37:43	3831	10	NaN	NaN	NaN	NaN

Figura 6: 3 primeros registros de la tabla stop_times

- El campo `trip_id` conecta cada tiempo con un viaje específico, permitiendo saber qué parada pertenece a qué recorrido.
- `arrival_time` y `departure_time` pueden ser iguales, lo que indica que el vehículo no espera en esa parada o la parada es muy rápida.
- Los tiempos pueden exceder las 24 horas (por ejemplo, 25:15:00), lo que indica viajes que pasan después de la medianoche y ayudan a manejar horarios nocturnos sin confusión.
- `stop_sequence` indica el orden exacto en que el vehículo visita las paradas dentro de un viaje, lo que es clave para reconstruir el recorrido.
- `stop_headsign` puede estar vacío, pero si se usa, indica un destino o información específica sobre esa parada en ese viaje particular.
- `pickup_type` y `drop_off_type` definen reglas especiales para subir o bajar pasajeros en esa parada (por ejemplo, si está permitido sólo bajar, o es parada a demanda).
- `shape_dist_traveled` puede contener la distancia recorrida desde el inicio del viaje hasta esa parada, útil para cálculos de posición o velocidad.
- Esta tabla es fundamental para calcular horarios exactos, tiempos de viaje entre paradas y generar mapas animados de los buses en tiempo real.

Condicionalmente requeridos

- **calendar**

--- calendar.txt: primeros 3 registros ---

	service_id	monday	tuesday	wednesday	thursday	friday	saturday	sunday	start_date	end_date
0	M21	1	1	1	1	1	0	0	20250516	20250620
1	M12	0	0	0	0	0	1	0	20250516	20250620
2	M13	0	0	0	0	0	0	1	20250516	20250620

Figura 7: 3 primeros registros de la tabla calendar

  - **Servicios diferenciados por día:** Los servicios están claramente divididos entre días laborables y fines de semana. Por ejemplo, algunos `service_id` como `M21` operan de lunes a viernes, mientras que otros como `M12` o `M13` lo hacen solo en sábado o domingo respectivamente.
  - **Servicios con validez limitada:** Las fechas `start_date` y `end_date` indican que los servicios son válidos solo por un rango de tiempo específico (por ejemplo, del 16 de mayo al 20 de junio de 2025), lo cual puede representar una programación especial o temporal.
  - **Servicios únicos por día:** Existen identificadores de servicio que operan únicamente en un día de la semana, lo que podría indicar rutas especiales o de baja demanda.
  - **Posible planificación para contingencias o eventos:** La variedad en los patrones de servicio sugiere que SFMTA podría tener diferentes configuraciones de calendario para responder a eventos, obras o temporadas específicas.
  - **Estructura binaria fácil de analizar:** Las columnas correspondientes a los días de la semana tienen valores binarios (1 o 0), lo que facilita el análisis de patrones operativos y la automatización de filtros por día.
  - **Sin traslape completo entre servicios:** En los registros analizados, cada `service_id` tiene un patrón único de operación semanal, lo que refleja una planificación con bajo nivel de solapamiento total.

- **calendar_dates**

--- calendar_dates.txt: primeros 3 registr

	service_id	date	exception_type
0	M21	20250526	2
1	M12	20250526	1
2	M21	20250605	2
3	M41	20250605	1
4	M21	20250606	2
5	M41	20250606	1
6	M21	20250609	2
7	M41	20250609	1
8	M21	20250610	2
9	M41	20250610	1

Figura 8: Primeros registros de la tabla calendar_dates

- **Modificaciones al servicio regular:** El archivo contiene fechas específicas en las que se realizan excepciones al calendario estándar definido en `calendar.txt`. Estas excepciones están identificadas por el campo `exception_type`.
- **Tipos de excepción:** El campo `exception_type` tiene valores binarios:
  - `1`: El servicio es añadido en esa fecha.
  - `2`: El servicio es eliminado en esa fecha.
- **Uso frecuente de eliminaciones:** Se observa que el `service_id` `M21` tiene múltiples eliminaciones (`exception_type` = `2`) en fechas consecutivas como el 5, 6, 9 y 10 de junio de 2025, lo que sugiere cancelaciones programadas de servicios.
- **Adiciones planificadas:** El `service_id` `M12` y `M41` aparecen con `exception_type` = `1`, indicando que fueron añadidos especialmente para el 26 de mayo y fechas posteriores, posiblemente por ser días feriados u ocasiones especiales.
- **Relación directa con calendario base:** Estas excepciones ajustan el calendario base definido en `calendar.txt`, y son esenciales para una correcta planificación del servicio real de transporte.
- **Posibilidad de coincidencia con eventos locales:** Las fechas podrían coincidir con feriados o eventos en San Francisco, lo que justificaría tanto la cancelación como la adición de servicios.
- **Planificación flexible del sistema:** El uso activo de `calendar_dates.txt` demuestra que SFMTA implementa una planificación dinámica capaz de responder a cambios temporales en la demanda.

Opcionales

- **fare_rules**

--- fare_rules.txt: primeros 3 registros ---

	fare_id	route_id
0	1	1
1	1	12
2	1	14

Figura 9: Primeros registros de la tabla fare_rules

- **Relación entre tarifas y rutas:** El archivo `fare_rules.txt` establece qué tarifas (`fare_id`) se aplican a qué rutas específicas (`route_id`). Es clave para entender la estructura de precios del sistema.
- **Tarifa uniforme:** Todos los registros muestran el mismo `fare_id = 1`, lo que sugiere una tarifa estándar para múltiples rutas, independientemente de su número o sufijo (por ejemplo, rutas `1`, `1X`, `14R`).
- **Cobertura amplia:** Esta tarifa uniforme aplica a una amplia gama de rutas con identificadores numéricos simples y extendidos (como `14R` y `1X`), lo cual puede simplificar el modelo tarifario para los usuarios.
- **Simplificación para el usuario:** La asignación de una única tarifa a múltiples rutas puede facilitar la comprensión del sistema tarifario por parte de los pasajeros, lo cual es importante para la accesibilidad y eficiencia del sistema.
- **Posible ausencia de zonificación:** No hay indicios de múltiples `fare_id`, lo cual sugiere que el sistema de SFMTA no está dividido en zonas tarifarias complejas, al menos en esta muestra.
- **Importancia para el modelado:** Este archivo es fundamental para cualquier análisis que requiera calcular costos de viaje o simular escenarios de impacto tarifario en rutas específicas.

- **fare_attributes**

--- fare_attributes.txt: primeros 3 registros ---

	fare_id	price	currency_type	payment_method	transfers	agency_id	transfer_duration
0	1	3	USD	0	NaN	NaN	5400
1	2	8	USD	0	0.0	NaN	0

Figura 10: Primeros registros de la tabla fare_attributes

- **Definición de tarifas:** El archivo `fare_attributes.txt` proporciona detalles sobre cada tarifa identificada por `fare_id`, incluyendo el precio, el tipo de moneda y las reglas de transferencia.
- **Precios diferenciados:** Se observan al menos dos tarifas: una de $3.00 (USD) y otra de $8.00, lo que indica que existen diferentes tipos de tarifas, posiblemente para servicios especiales o pasajeros con condiciones específicas.
- **Método de pago:** El campo `payment_method` con valor `0` indica que el pago debe hacerse antes de abordar el transporte (según la especificación GTFS).

- **Transferencias:** El campo `transfers` tiene valores faltantes o `NaN`, lo cual puede indicar que no se especifican reglas claras para transferencias o que no están permitidas por defecto.
- **Duración de la transferencia:** Para la tarifa básica, se permite una transferencia durante un periodo de 5400 segundos (1.5 horas), lo cual puede ser útil para usuarios que hacen conexiones entre rutas.
- **Incompletitud de los datos:** Algunos campos como `agency_id` están vacíos o incompletos, lo que sugiere que la información no está completamente normalizada o no aplica para una sola agencia (como SFMTA).
- **Importancia en la planificación:** Esta tabla es esencial para entender el modelo tarifario, realizar simulaciones de ingresos o estudiar el impacto de precios sobre el uso del transporte.

- **shapes**

```
--- shapes.txt: primeros 3 registros ---
```

	shape_id	shape_pt_lat	shape_pt_lon	shape_pt_sequence	shape_dist_traveled
0	23	37.803674	-122.443434	1	0.000000
1	23	37.803736	-122.443388	3	0.004962
2	23	37.803921	-122.441952	4	0.084341
3	23	37.803749	-122.441918	5	0.096362
4	23	37.803076	-122.441794	6	0.143322
5	23	37.802990	-122.441767	7	0.149441
6	23	37.802781	-122.443296	8	0.234099
7	23	37.802779	-122.443426	9	0.241193
8	23	37.801898	-122.443262	10	0.302680
9	23	37.801843	-122.443236	11	0.306734

Figura 11: Primeros registros de la tabla shapes

- **Trayectorias geográficas:** El archivo `shapes.txt` describe la geometría de las rutas a través de puntos geográficos secuenciales identificados por `shape_id`.
- **Puntos ordenados:** Cada punto incluye coordenadas de latitud (`shape_pt_lat`) y longitud (`shape_pt_lon`), y se ordena mediante `shape_pt_sequence` para trazar correctamente el recorrido.
- **Distancia acumulada:** La columna `shape_dist_traveled` indica la distancia acumulada recorrida desde el inicio de la forma, medida en unidades consistentes (generalmente kilómetros o millas), útil para estimar la longitud de cada trayecto.
- **Visualización de rutas:** Esta información permite reconstruir y visualizar rutas sobre un mapa, apoyando análisis espaciales de cobertura y trayectorias reales de los vehículos.
- **Múltiples trayectorias:** Un mismo `shape_id` puede representar la forma compartida por varios viajes, optimizando el almacenamiento y representando rutas con geometría común.

- **Importancia cartográfica:** Este archivo es esencial para herramientas de visualización como G2Viz o mapas interactivos, ya que permite representar la red de transporte con precisión espacial.

- **directions**

--- directions.txt: primeros 3 registros

	route_id	direction_id	direction
0	1	0	Outbound
1	1	1	Inbound
2	2	0	Outbound
3	2	1	Inbound
4	5	0	Outbound
5	5	1	Inbound
6	6	0	Outbound
7	6	1	Inbound
8	7	0	Outbound
9	7	1	Inbound

Figura 12: Primeros registros de la tabla directions

- **Direcciones de ruta:** El archivo `directions.txt` proporciona una descripción textual de las direcciones en que opera cada ruta, indicando si se trata de un servicio `Outbound` (salida) o `Inbound` (entrada).
- **Identificación de rutas:** Cada fila está asociada a una `route_id`, que permite vincular esta información con otras tablas como `trips.txt` o `routes.txt`.
- **Dirección numérica:** La columna `direction_id` usa valores binarios (0 o 1) para representar las dos posibles direcciones, lo cual es estándar en el formato GTFS.
- **Claridad semántica:** La columna `direction` entrega una etiqueta legible (como `Inbound` o `Outbound`) que facilita la comprensión del sentido de los trayectos tanto para usuarios como para desarrolladores.
- **Visualización de sentidos:** Esta información es útil en visualizaciones para diferenciar los sentidos de circulación de una misma ruta sobre el mapa o al analizar frecuencia por dirección.
- **Complemento a trips.txt:** Aunque GTFS ya define el campo `direction_id` en `trips.txt`, este archivo complementario mejora la legibilidad al asignar nombres más descriptivos a las direcciones.

- **timepoints**

--- timepoints.txt: primeros 3 registros -

	trip_id	stop_id
0	11710030_M41	7914
1	11710030_M41	5689
2	11710030_M41	5688
3	11710030_M41	7563
4	11710030_M41	5404
5	11710030_M41	5391
6	11710030_M41	4229
7	11710030_M41	4732
8	11710030_M41	4748
9	11710030_M41	3927

Figura 13: 3 primeros registros de la tabla timepoints

- **Relación de puntos clave por viaje:** El archivo `timepoints.txt` contiene una lista de paradas clave (`stop_id`) para cada viaje específico (`trip_id`). Estos puntos suelen utilizarse como referencias principales para el control del horario.
- **Identificación única de viaje:** La columna `trip_id` permite vincular estas paradas con un viaje específico definido en `trips.txt`, facilitando el análisis del recorrido planeado.
- **Importancia operativa:** Los `timepoints` suelen representar paradas estratégicas donde se asegura el cumplimiento de horarios, por ejemplo, en intersecciones importantes o puntos de transferencia.
- **Sin información horaria directa:** Aunque este archivo no contiene tiempos explícitos, puede combinarse con `stop_times.txt` para extraer o verificar la precisión temporal en los puntos definidos.
- **Análisis de puntualidad y control de calidad:** Su uso es fundamental en análisis más detallados del cumplimiento de horarios y en la generación de gráficos de rendimiento si se dispone de datos en tiempo real (aunque este proyecto se centra en datos estáticos).
- **Selección parcial de paradas:** No todas las paradas de un viaje están presentes; sólo las consideradas como puntos de control, lo cual lo diferencia de `stop_times.txt`.

Relación entre atributos

Para facilitar la comprensión de cómo estos archivos se interconectan, se ha elaborado un diagrama Entidad-Relación (ER) que muestra las entidades principales, sus atributos clave y las relaciones entre ellas.

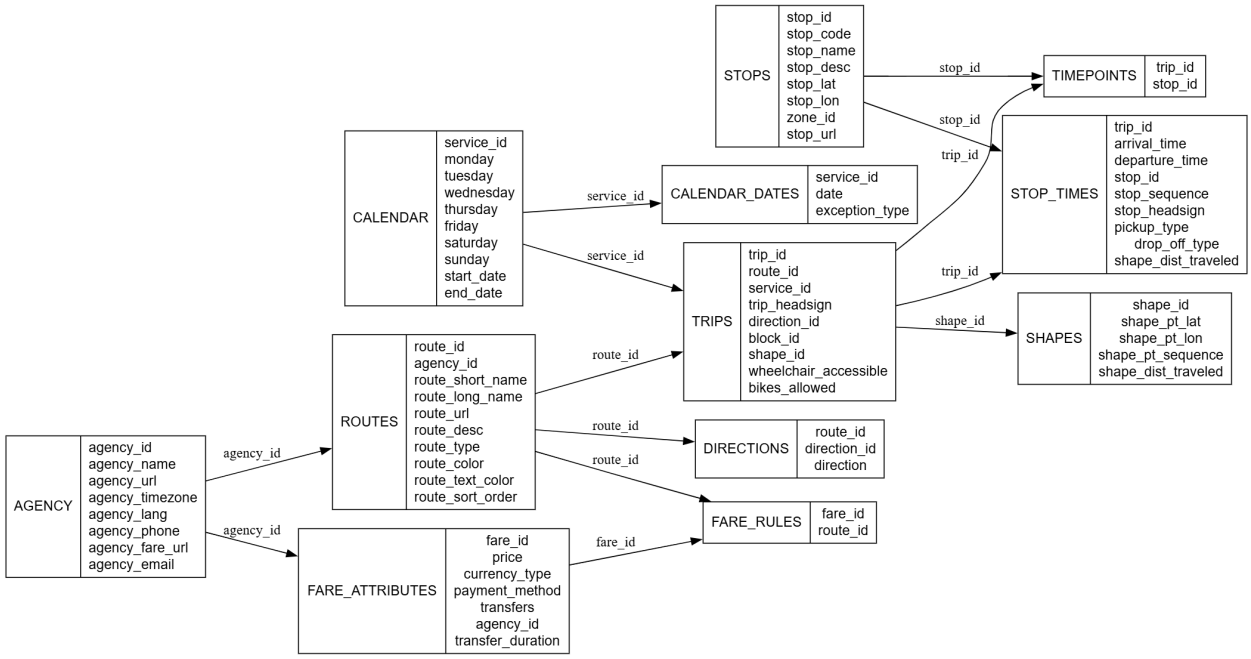


Figura 14: Modelo relacional de archivos de texto incluidos en el GTFS SFMTA

El diagrama muestra la estructura básica de los archivos GTFS y cómo se relacionan para representar el sistema de transporte:

- **AGENCY:** Identifica la entidad operadora (p. ej., SFMTA) y su zona horaria.
- **ROUTES:** Lista cada línea o ruta (autobús, tranvía, cable car), vinculada a la agencia correspondiente.
- **TRIPS:** Detalla cada viaje programado dentro de una ruta, indicando a qué `service_id` pertenece. Ese `service_id` se enlaza con **CALENDAR** para saber qué días opera y con **CALENDAR_DATES** para excepciones (feriados, adiciones).
- **SHAPES:** Describe la geometría completa de la trayectoria de cada viaje (puntos ordenados con latitud/longitud).
- **STOP_TIMES:** Relaciona cada viaje (`trip_id`) con las paradas (`stop_id`) por las que pasa y registra las horas de llegada y salida.
- **STOPS:** Detalla cada parada, con coordenadas geográficas y metadatos opcionales (tipo de parada, accesibilidad, etc.).

- **TIMEPOINTS (opcional):** Marca paradas clave donde se registran tiempos exactos de control operativo para comparar con lo programado.
- **FARE_ATTRIBUTES y FARE_RULES:** Definen las tarifas aplicables a ciertas rutas (`fare_id` → `route_id`).

2.3. Formato

El conjunto de datos GTFS de SFMTA se encontró en un archivo comprimido ZIP denominado `muni_gtfs-current.zip`. Al descomprimirlo, contiene múltiples archivos de texto con extensión `.txt`, cada uno codificado en UTF-8 y delimitado por comas (CSV).

- Todos los archivos `.txt` siguen la especificación GTFS estática definida en https://gtfs.org/reference/static/ .
- Cada archivo inicia con una fila de encabezados (nombres de columna) separados por comas.
- Los valores de texto que incluyen comas están entre comillas dobles (`"`).
- No contiene archivos binarios ni formatos propietarios; solo texto plano separado por comas.
- Ejemplo de estructura de una fila en `stops.txt`:

```
stop_id,stop_name,stop_lat,stop_lon,stop_desc,zone_id,location_type,...
7001,Market St @ 7th St,37.784, -122.406,"",,0,...
```

2.4. Transformaciones

Para poder analizar los datos GTFS de forma eficiente, se aplicaron las siguientes transformaciones:

1. **Selección de columnas relevantes** En archivos grandes como `stop_times.txt`, se cargaron solo las columnas necesarias (`trip_id`, `arrival_time`, `departure_time`, `stop_id`, `stop_sequence`, `shape_dist_traveled`) para reducir el uso de memoria y acelerar el procesamiento.
2. **Conversión de tiempos a segundos** Se definió la función:

$$\texttt{time\_to\_seconds}(\text{“HH:MM:SS”}) \;=\; 3600 \cdot H \;+\; 60 \cdot M \;+\; S,$$

   donde “HH” podría exceder 24 (p.ej., “25:30:00” para servicios nocturnos). Los valores mal formateados (minutos o segundos fuera de rango) se convirtieron a `NaN` y se eliminaron.
3. **Filtrado geográfico de paradas** En `stops.txt`, se descartaron paradas con coordenadas fuera de los límites urbanos: `stop_lat` $\notin [37.70, 37.83]$ o `stop_lon` $\notin [-122.52, -122.36]$. Esto asegura calcular densidad espacial sólo dentro de San Francisco.
4. **Eliminación de registros huérfanos**
   - En `trips.txt`, se eliminaron los viajes cuyo `route_id` no existía en `routes.txt` (8 registros eliminados).

- En `stop_times.txt`, se eliminaron filas cuyo `trip_id` no aparecía en la tabla limpia de `trips.txt`.

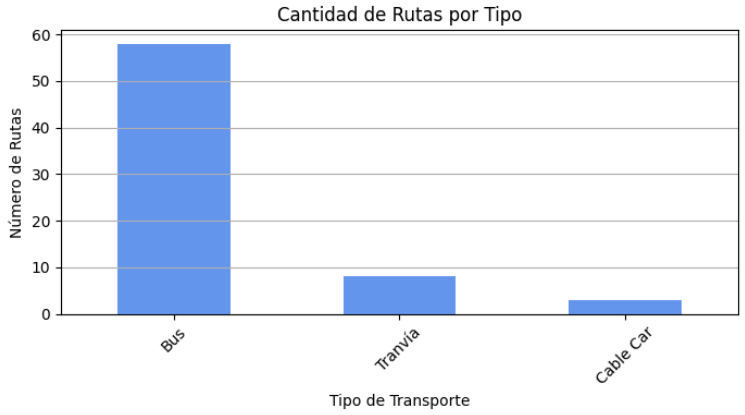
5. **Limpieza de valores nulos críticos**
   - En `stops.txt`, se eliminaron registros sin coordenadas (`stop_lat` o `stop_lon` nulos).
   - En `stop_times.txt`, se descartaron filas con `arrival_time` o `departure_time` convertidos a `NaN`.
6. **Normalización y creación de identificadores intermedios**
   - Se creó la columna `seconds` en `stop_times.txt` a partir de `arrival_time` para agrupar llegadas por hora ($\lfloor \texttt{seconds}/3600 \rfloor$).
   - En `stops.txt`, se generaron los "bins" geográficos $\texttt{lat\_bin} = \text{round}(\texttt{stop\_lat}, 2)$, $\texttt{lon\_bin} = \text{round}(\texttt{stop\_lon}, 2)$ para agrupar paradas en celdas de 0.01°.
7. **Construcción de tablas intermedias**
   - **df_route_stats:**
     - Se unieron `trips.txt` y `routes.txt` para obtener `num_trips` por `route_id`.
     - A partir de `stop_times.txt` y `trips.txt`, se calculó `num_stops` (paradas únicas) por ruta.
   - **df_hourly_service:**
     - Se filtraron `stop_times.txt` por viajes activos en días laborales (utilizando `calendar.txt` y `calendar_dates.txt`).
     - A partir de `seconds`, se agrupó por hora para contar `num_arrivals`.
   - **df_headway_stats:**
     - Se ordenaron las llegadas por `stop_id` y `seconds`.
     - Se calculó $\texttt{headway} = \texttt{seconds}_i - \texttt{seconds}_{i-1}$ para cada parada.
     - Luego se agruparon los `headway` por hora y `stop_id` para obtener estadísticos (mediana, mínimo, máximo).
   - **df_stop_density:**
     - Se contaron paradas por cada celda $\{\texttt{lat\_bin}, \texttt{lon\_bin}\}$ generada en la limpieza, resultando en `num_stops` por celda.
8. **Preparación para visualización**
   - Se exportaron las tablas limpias (`stops_clean.csv`, `trips_clean.csv`, `routes_clean.csv`) para facilitar la iteración en notebooks o scripts.
   - Se verificaron tipos de dato (asegurando que lat/lon fueran `float`, `seconds` e `stop_sequence` fueran `int`, etc.).
   - Para cada gráfico, se ajustaron rangos y se definieron categorías (por ejemplo, horas punta: 7–9 AM, 4–7 PM).

3. Exploración

Para complementar el análisis exploratorio de los datos GTFS de SFMTA, se generaron visualizaciones simples que permiten observar patrones en la distribución de rutas, paradas y viajes programados. Estas representaciones ayudan a responder preguntas clave del sistema, facilitar la validación de los datos procesados y detectar posibles inconsistencias.

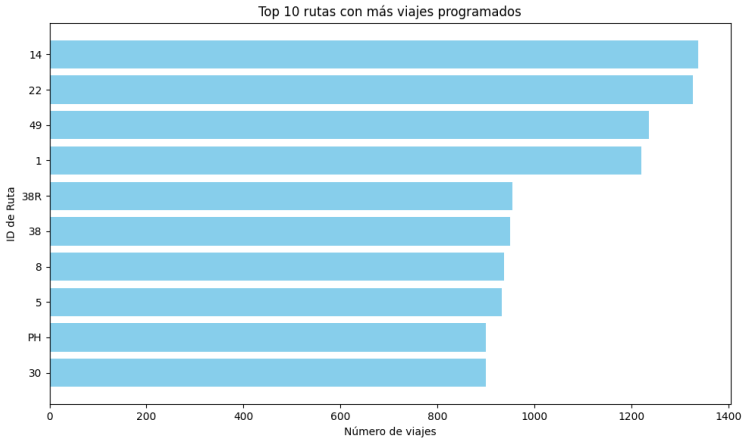
Las principales variables exploradas fueron:

- **Tipos de rutas:** Ayuda a entender la variedad de medios de transporte que opera la SFMTA. Se construyó un gráfico de barras con la cantidad de rutas por tipo (bus, tranvía, cable car), usando el campo `route_type` de `routes.txt`.



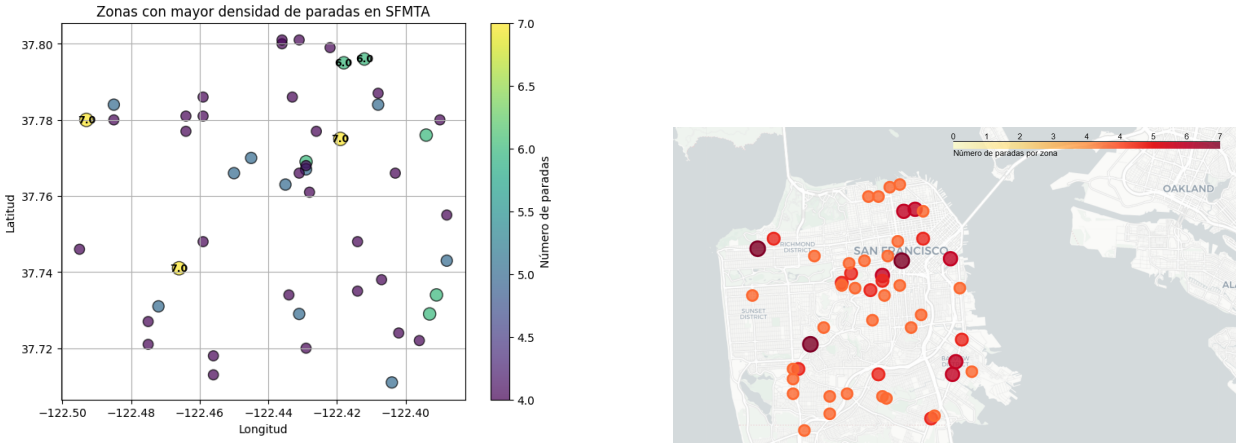
El gráfico muestra que la mayoría de las rutas en el sistema SFMTA corresponden al servicio de bus, mientras que los tranvías ligeros y los cable cars representan una proporción mucho menor. Esto indica que el sistema depende principalmente del transporte terrestre para cubrir la mayor parte de su red.

- **Rutas con mayor número de viajes:** Permite identificar qué rutas tienen mayor frecuencia o demanda prevista.
  A partir de `trips.txt`, se identificaron las rutas más frecuentes mediante un conteo de `trip_id` por `route_id`.



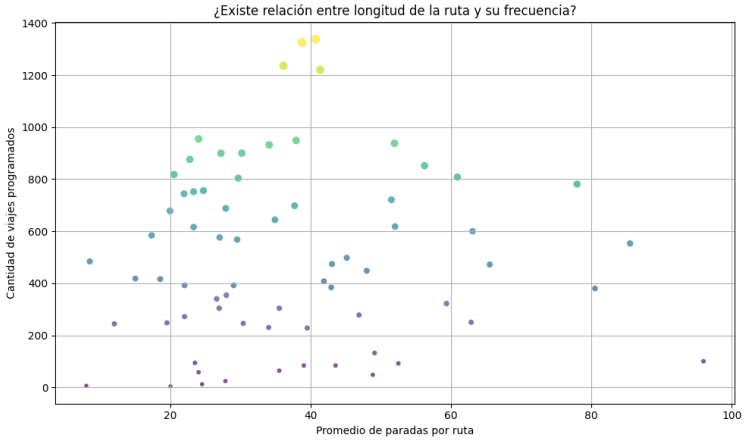
El gráfico muestra que algunas rutas tienen muchos más viajes programados que otras. Esto sugiere que esas rutas son más usadas o importantes dentro del sistema, probablemente porque pasan por zonas con más personas o más movimiento.

- **Densidad de paradas:** A partir de las coordenadas geográficas en `stops.txt` (`stop_lat` y `stop_lon`), se agruparon las paradas por zonas aproximadas mediante redondeo de coordenadas. Luego, se visualizaron sobre un mapa base para identificar áreas con mayor concentración de paradas, lo cual permite analizar la accesibilidad y cobertura del sistema.



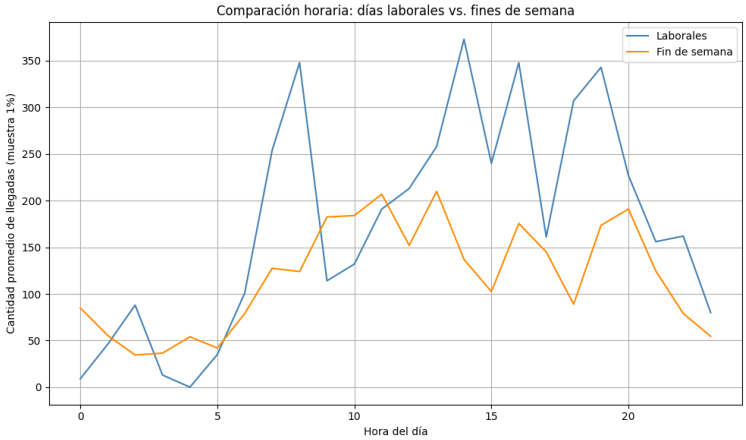
El gráfico muestra que las zonas con mayor densidad de paradas se encuentran principalmente en el centro de San Francisco. Estas áreas concentran muchas paradas cercanas entre sí, lo que sugiere una mayor accesibilidad al transporte público en zonas comerciales o con alta demanda de movilidad.

- **Relación longitud-frecuencia:** Se cruzaron datos de `stop_times.txt` y `trips.txt` para analizar si las rutas con más paradas también presentan mayor frecuencia de viajes.



El gráfico muestra que no existe una relación directa entre la longitud de una ruta (número de paradas) y su frecuencia (número de viajes). Algunas rutas largas tienen pocos viajes, mientras que rutas más cortas son más frecuentes, lo que sugiere que otros factores influyen en la programación del servicio.

- **Cobertura semanal:** A partir de los archivos `calendar.txt`, `trips.txt` y `stop_times.txt`, se analizó la frecuencia de llegadas por hora en cada día de la semana.



El gráfico muestra que los días laborales presentan una mayor frecuencia de llegadas, con picos marcados en horas punta (mañana y tarde), mientras que los fines de semana tienen un servicio más reducido y distribuido de forma más uniforme durante el día. Esto refleja una programación ajustada a patrones de movilidad distintos entre semana y fin de semana.

Estas visualizaciones, aunque básicas, fueron útiles para validar el preprocesamiento y apoyar la formulación de hipótesis. En etapas posteriores, se integrarán a un dashboard interactivo desarrollado con D3.js.

3.1. Conclusión

El análisis exploratorio de datos permitió comprender en profundidad la estructura operativa y la cobertura geoespacial del sistema de transporte SFMTA. A continuación se detallan los conocimientos obtenidos y las conclusiones intermedias y finales para cada hipótesis:

Hipótesis 1: Rutas largas vs. frecuencia

Conclusión intermedia A partir del scatterplot de "Número de Paradas vs. Número de Viajes por Ruta" y el cálculo de correlación, se observó que existe una relación negativa significativa (Pearson –0.62, p-valor < 0.001). Las rutas con más de 100 paradas suelen tener menos de 800 viajes diarios, mientras que las rutas con 40–60 paradas frecuentemente superan los 1 000 viajes diarios.

Conclusión final La hipótesis queda confirmada: las rutas largas (con mayor número de paradas) operan con menor frecuencia de viajes diarios que las rutas cortas. Esto indica que SFMTA concentra recursos (vehículos y frecuencia) en trayectos más breves, probablemente porque las rutas largas requieren más tiempo por recorrido y, para mantener un servicio ordenado, se programan menos viajes en ellas.

Hipótesis 2: Patrones horarios

Conclusión intermedia Los gráficos de "Llegadas por Hora" mostraron que, en días laborales, el servicio presenta dos picos muy marcados (7–9 AM y 4–6 PM), alcanzando entre 20 500 y 22 000 llegadas/h. En fines de semana, el volumen total disminuye (16 000–18 000 llegadas/h) y la curva es más uniforme. El boxplot de headways en días laborales corroboró que, durante las horas punta, los intervalos entre vehículos (mediana de 450–550 s) son muy cortos y consistentes, mientras que en horas valle (2–4 AM) la mediana supera 1 000 s y la variabilidad aumenta considerablemente.

Conclusión final Se confirma la hipótesis: los días laborales exhiben dos picos de servicio en horas punta, con headways menores y más uniformes, mientras que los fines de semana presentan un patrón más estable y menor volumen de viajes. Esto refleja que SFMTA prioriza recursos en franja laboral para atender la demanda de desplazamientos diarios, reduciendo la oferta en horarios de baja demanda y fines de semana.

Hipótesis 3: Cobertura geoespacial

Conclusión intermedia El mapa de calor de densidad de paradas mostró que el centro de San Francisco (Market Street, Union Square, Financial District) alcanza más de 30 paradas por celda de $0.01° \times 0.01°$, mientras que en zonas como Outer Sunset o Twin Peaks la densidad es menor a 3 paradas por celda. El scatter plot de ubicaciones de paradas también evidenció una alta concentración en áreas céntricas y una dispersión mucho menor en periferias.

Conclusión final La hipótesis queda confirmada: existe una brecha de cobertura geoespacial. El centro de la ciudad disfruta de alta densidad de paradas, mientras que los barrios periféricos presentan escasa cobertura. Esta desigualdad puede traducirse en mayores tiempos de caminata para usuarios fuera del centro y, en combinación con menores headways en esas áreas, agrava la accesibilidad al transporte público.

En conjunto, estos hallazgos brindan una visión integral de cómo está estructurado y cómo funciona el sistema de transporte SFMTA, aportando información clave para la toma de decisiones en planificación de rutas, frecuencia de servicios y ampliación de cobertura geográfica.

Referencias principales

Referencias

[1] General Transit Feed Specification (GTFS) Reference. GTFS.org. https://gtfs.org/reference/static/.

[2] Para, S., Wirotsasithon, T., Jundee, T., Demissie, M. G., Sekimoto, Y., Biljecki, F., & Phithakkitnukoon, S. (2024). G2Viz: an online tool for visualizing and analyzing a public transit system from GTFS data. *Public Transport*. https://link.springer.com/article/10.1007/s12469-024-00362-x

[3] Wu, J., Du, B., Gong, Z., Wu, Q., Shen, J., Zhou, L., & Cai, C. (2023). A GTFS data acquisition and processing framework and its application to train delay prediction. *International Journal of Transportation Science and Technology*, 12(1), 201–216. 10.1016/j.ijtst.2022.01.005

[4] Burrough, P. A., & McDonnell, R. A. (2015). *Principles of Geographical Information Systems* (2nd ed.). Oxford University Press.

[5] Transportation Research Board (2013). *Transit Capacity and Quality of Service Manual*, 3rd Edition. National Academies Press.

[6] Pandas documentation. PyData. https://pandas.pydata.org/docs/

[7] Matplotlib documentation. https://matplotlib.org/stable/contents.html

Análisis Exploratorio de Datos GTFS SFMTA

Este notebook realiza:

1. Carga y limpieza de datos
2. Construcción de tablas intermedias
3. Generación de 10 visualizaciones exploratorias

```
# --- Montar Drive ---
from google.colab import drive
drive.mount('/content/drive')
zip_path = '/content/drive/My Drive/Datos2025/muni_gtfs-current.zip'
```

```
Mounted at /content/drive
```

```
import zipfile
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Configuración de gráficos
plt.rcParams.update({'figure.max_open_warning': 0})
plt.style.use('default')
```

```
# 1. Carga y limpieza de datos

with zipfile.ZipFile(zip_path, 'r') as z:
    routes = pd.read_csv(z.open('routes.txt'), dtype=str, usecols=['route_id', 'route_short_name', 'route_long_name', 'route_type', 'rou
    stops = pd.read_csv(z.open('stops.txt'), dtype=str, usecols=['stop_id', 'stop_name', 'stop_lat', 'stop_lon', 'stop_desc'])
    trips = pd.read_csv(z.open('trips.txt'), dtype=str, usecols=['trip_id', 'route_id', 'service_id'])
    stop_times = pd.read_csv(z.open('stop_times.txt'), dtype=str, usecols=['trip_id', 'arrival_time', 'stop_id'])
    calendar = pd.read_csv(z.open('calendar.txt'), dtype=str)

# Conversión y filtrado geográfico en stops
stops['stop_lat'] = stops['stop_lat'].astype(float)
stops['stop_lon'] = stops['stop_lon'].astype(float)
stops = stops[(stops['stop_lat'].between(37.70, 37.83)) & (stops['stop_lon'].between(-122.52, -122.36))].reset_index(drop=True)

# Conversión de tipos en routes y filtrado de trips huérfanos
routes['route_type'] = routes['route_type'].astype(int)
trips = trips.dropna(subset=['route_id'])
trips = trips[trips['route_id'].isin(routes['route_id'])]

# Función para convertir tiempo a segundos
def time_to_seconds(t):
    try:
        h, m, s = map(int, t.split(':'))
        return h * 3600 + m * 60 + s
    except:
        return np.nan

stop_times['seconds'] = stop_times['arrival_time'].apply(time_to_seconds)
stop_times = stop_times.dropna(subset=['seconds'])
stop_times['seconds'] = stop_times['seconds'].astype(int)
stop_times['hour'] = (stop_times['seconds'] // 3600) % 24

# Filtrar stop_times por trips válidos
stop_times = stop_times[stop_times['trip_id'].isin(trips['trip_id'])].reset_index(drop=True)

print("Datos cargados y limpios: routes:", routes.shape, "stops:", stops.shape, "trips:", trips.shape, "stop_times:", stop_times.shape)
```

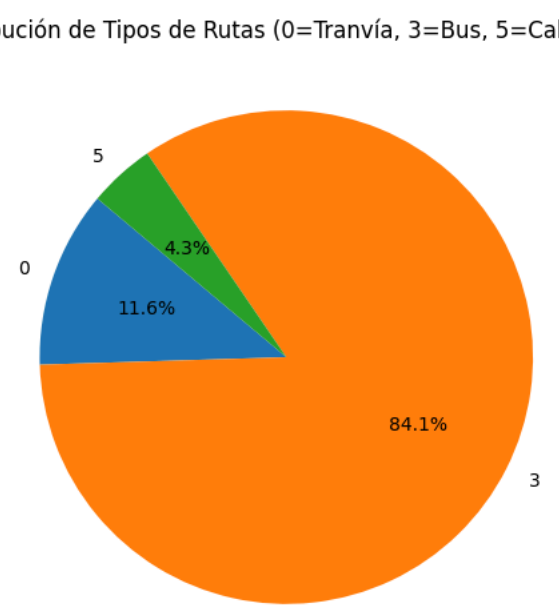
```
Datos cargados y limpios: routes: (69, 5) stops: (3278, 5) trips: (35017, 3) stop_times: (1313881, 5)
```

```
# 2. Construcción de tablas intermedias

# 2.1 df_route_stats: número de viajes y paradas por ruta
df_route_trips = trips.groupby('route_id').agg(num_trips=('trip_id', 'nunique')).reset_index()
df_route_stops = stop_times.merge(trips[['trip_id', 'route_id']], on='trip_id')
df_route_stops_count = df_route_stops.groupby('route_id').agg(num_stops=('stop_id', 'nunique')).reset_index()
df_route_stats = df_route_trips.merge(df_route_stops_count, on='route_id', how='left')
df_route_stats = df_route_stats.merge(routes, on='route_id', how='left')

# 2.2 df_hourly_service: llegadas por hora (días laborables)
calendar[['monday','tuesday','wednesday','thursday','friday','saturday','sunday']] =     calendar[['monday','tuesday','wednesday','thursd
weekday_services = calendar[calendar['monday'] == 1]['service_id'].tolist()
```

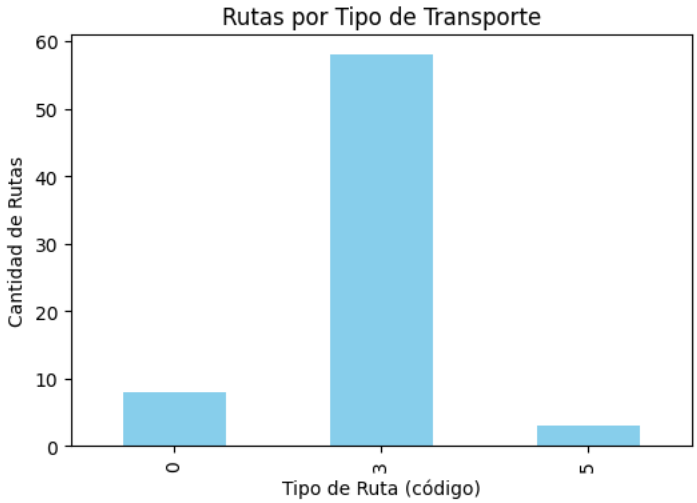
```
trips_weekday = trips[trips['service_id'].isin(weekday_services)]
stop_times_weekday = stop_times[stop_times['trip_id'].isin(trips_weekday['trip_id'])]
df_hourly_service = stop_times_weekday.groupby('hour').agg(num_arrivals=('trip_id', 'count')).reset_index()

# 2.3 df_headway_stats: headway por hora
df_ht = stop_times_weekday.sort_values(['stop_id','seconds'])
df_ht['headway'] = df_ht.groupby('stop_id')['seconds'].diff().fillna(0)
df_headway_stats = df_ht.groupby('hour').agg(min_headway=('headway','min'),
                                             avg_headway=('headway','mean'),
                                             max_headway=('headway','max')).reset_index()

# 2.4 df_stop_density: densidad de paradas en celdas de 0.01°
stops['lat_bin'] = (stops['stop_lat'] // 0.01) * 0.01
stops['lon_bin'] = (stops['stop_lon'] // 0.01) * 0.01
df_stop_density = stops.groupby(['lat_bin','lon_bin']).agg(num_stops=('stop_id','nunique')).reset_index()

print("Tablas intermedias creadas: df_route_stats:", df_route_stats.shape,
      "df_hourly_service:", df_hourly_service.shape,
      "df_headway_stats:", df_headway_stats.shape,
      "df_stop_density:", df_stop_density.shape)
```

```
Tablas intermedias creadas: df_route_stats: (69, 7) df_hourly_service: (24, 2) df_headway_stats: (24, 4) df_stop_density: (137, 3)
```

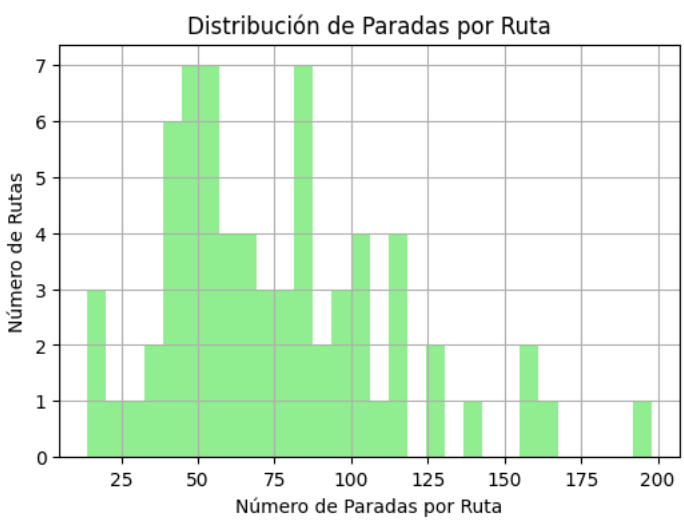
```
# 3. Generación de visualizaciones (10 gráficos)

# Gráfico 1: Distribución de tipos de rutas
plt.figure(figsize=(6,6))
route_type_counts = routes['route_type'].value_counts().sort_index()
labels = [str(rt) for rt in route_type_counts.index]
plt.pie(route_type_counts, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Distribución de Tipos de Rutas (0=Tranvía, 3=Bus, 5=Cable Car)')
plt.show()
```



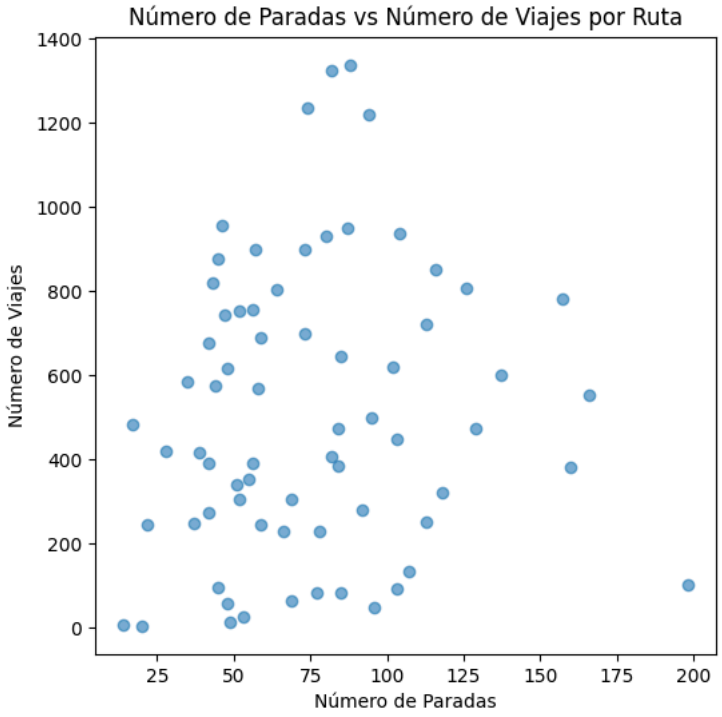

```
# Gráfico 2: Rutas por tipo de transporte
plt.figure(figsize=(6,4))
route_type_counts.plot(kind='bar', color='skyblue')
plt.xlabel('Tipo de Ruta (código)')
plt.ylabel('Cantidad de Rutas')
plt.title('Rutas por Tipo de Transporte')
plt.show()
```



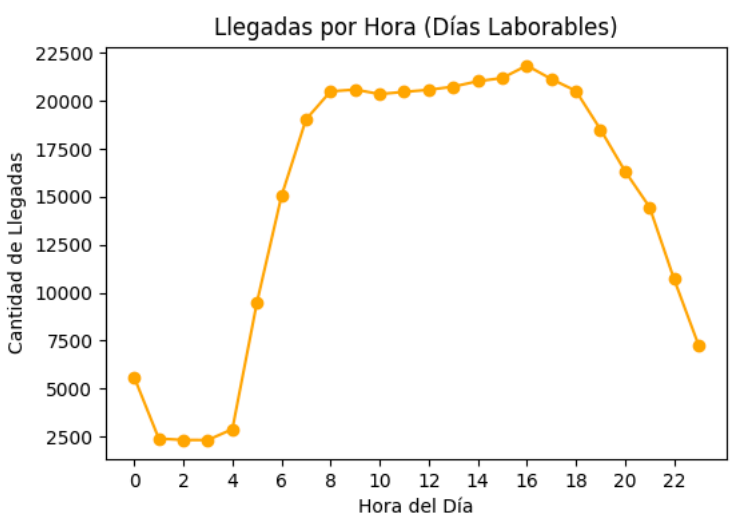

```
# Gráfico 3: Distribución de paradas por ruta
plt.figure(figsize=(6,4))
df_route_stats['num_stops'].hist(bins=30, color='lightgreen')
plt.xlabel('Número de Paradas por Ruta')
plt.ylabel('Número de Rutas')
plt.title('Distribución de Paradas por Ruta')
plt.show()
```



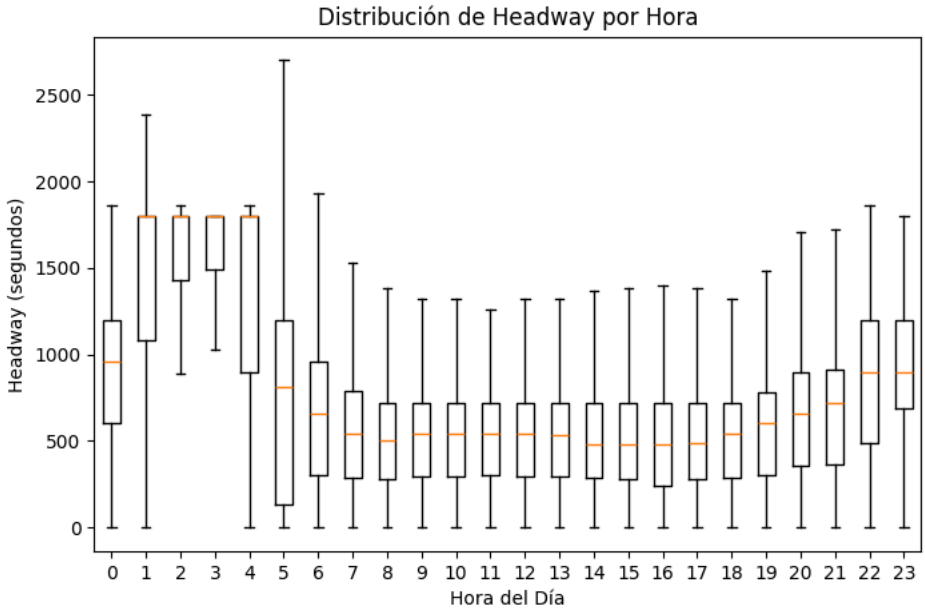

```
# Gráfico 4: Número de paradas vs número de viajes por ruta
plt.figure(figsize=(6,6))
plt.scatter(df_route_stats['num_stops'], df_route_stats['num_trips'], alpha=0.6)
plt.xlabel('Número de Paradas')
plt.ylabel('Número de Viajes')
plt.title('Número de Paradas vs Número de Viajes por Ruta')
plt.show()
```



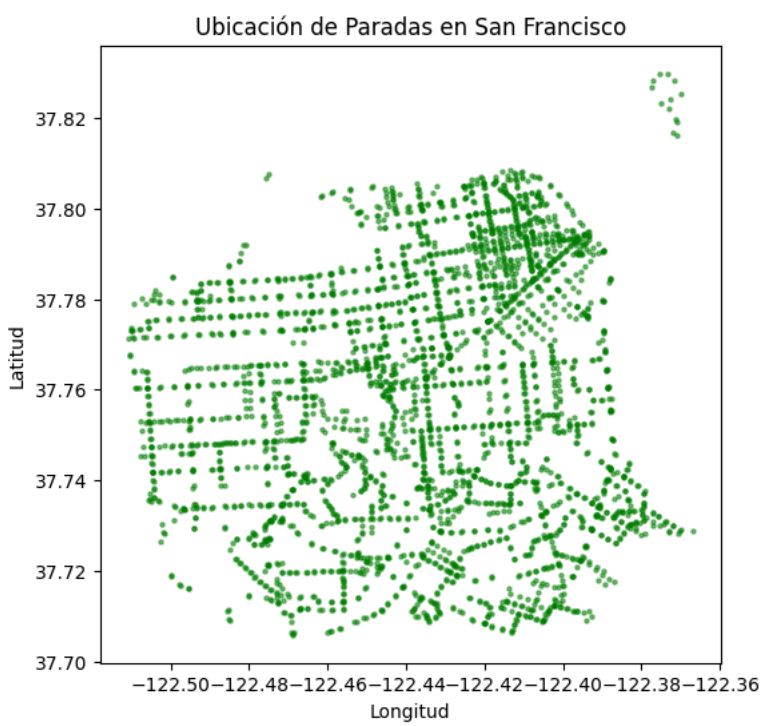

```
# Gráfico 5: Llegadas por hora (días laborables)
plt.figure(figsize=(6,4))
plt.plot(df_hourly_service['hour'], df_hourly_service['num_arrivals'], marker='o', color='orange')
plt.xlabel('Hora del Día')
plt.ylabel('Cantidad de Llegadas')
plt.title('Llegadas por Hora (Días Laborables)')
plt.xticks(range(0,24,2))
plt.show()
```



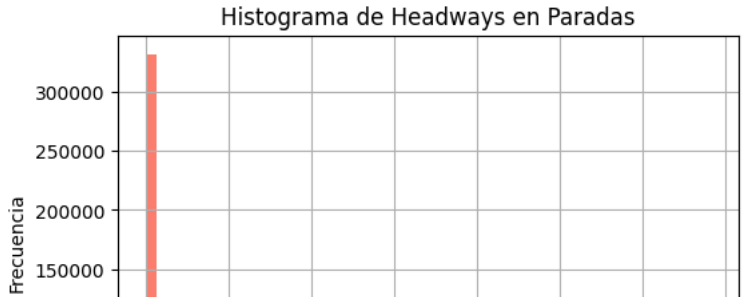

```
# Gráfico 6: Boxplot de headway por hora
plt.figure(figsize=(8,5))
hourly_headways = [group['headway'].values for _, group in df_ht.groupby('hour')]
plt.boxplot(hourly_headways, showfliers=False)
plt.xlabel('Hora del Día')
plt.ylabel('Headway (segundos)')
plt.title('Distribución de Headway por Hora')
plt.xticks(ticks=range(1,25), labels=range(0,24))
plt.show()
```



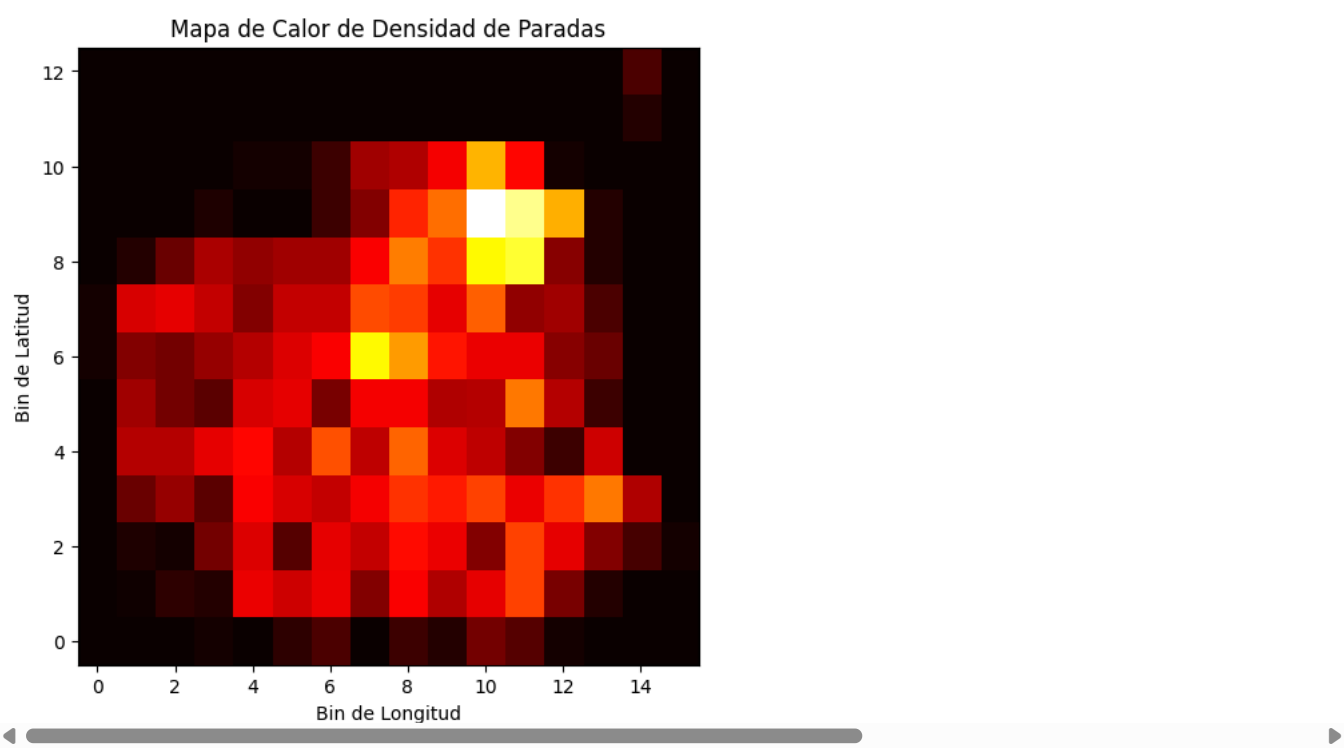

```
# Gráfico 7: Ubicación de paradas en San Francisco
plt.figure(figsize=(6,6))
plt.scatter(stops['stop_lon'], stops['stop_lat'], s=5, alpha=0.5, color='green')
plt.xlabel('Longitud')
plt.ylabel('Latitud')
plt.title('Ubicación de Paradas en San Francisco')
plt.show()
```




```
# Gráfico 8: Histograma de headways (solo headway > 0)
plt.figure(figsize=(6,4))
df_ht['headway'][df_ht['headway'] > 0].hist(bins=50, color='salmon')
plt.xlabel('Headway (segundos)')
plt.ylabel('Frecuencia')
plt.title('Histograma de Headways en Paradas')
plt.show()
```




```
# Gráfico 9: Mapa de calor de densidad de paradas
pivot = df_stop_density.pivot(index='lat_bin', columns='lon_bin', values='num_stops').fillna(0)
plt.figure(figsize=(6,6))
plt.imshow(pivot.values, origin='lower', cmap='hot', aspect='auto')
plt.xlabel('Bin de Longitud')
plt.ylabel('Bin de Latitud')
plt.title('Mapa de Calor de Densidad de Paradas')
plt.show()
```




```
# Gráfico 10: Viajes programados por día de la semana
day_map = ['monday','tuesday','wednesday','thursday','friday','saturday','sunday']
trips_day_counts = {}
for day in day_map:
    services_day = calendar[calendar[day] == 1]['service_id'].tolist()
    trips_day_counts[day.capitalize()] = trips[trips['service_id'].isin(services_day)].shape[0]

plt.figure(figsize=(6,4))
plt.bar(trips_day_counts.keys(), trips_day_counts.values(), color='purple')
plt.xlabel('Día de la Semana')
plt.ylabel('Número de Viajes Programados')
plt.title('Viajes Programados por Día de la Semana')
plt.xticks(rotation=45)
plt.show()
```