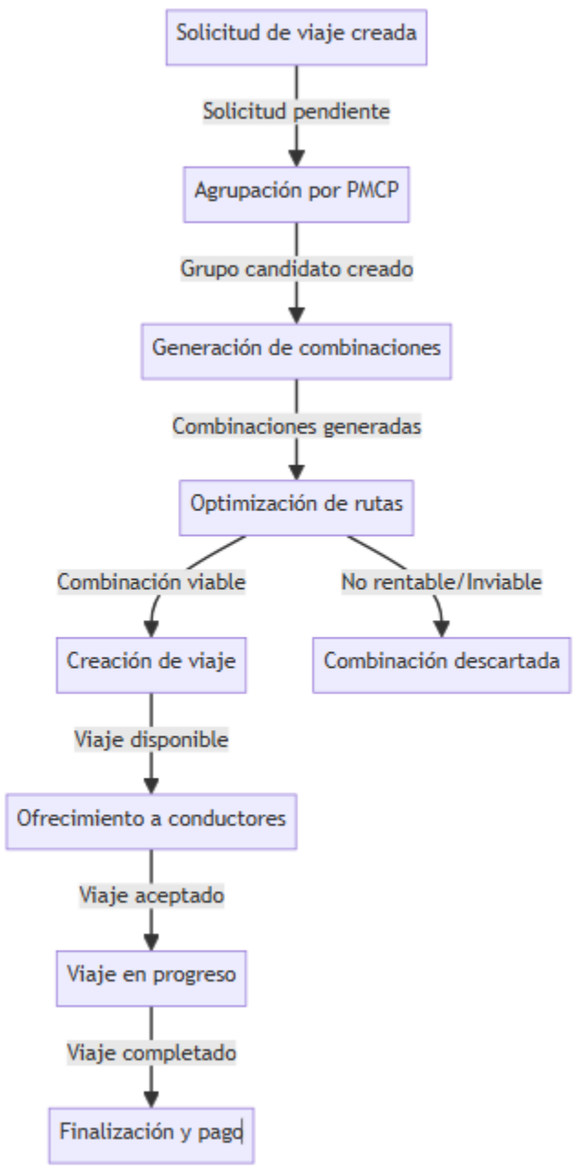
DOCUMENTACION GENERAL

Flujo de Procesamiento

1) **Solicitud de viaje:** El pasajero crea una solicitud con origen y/o destino en un PMCP.
2) **Agrupación:** Las solicitudes se agrupan por PMCP común.
3) **Combinaciones:** Se generan combinaciones viables de pasajeros.
4) **Optimización:** Se optimizan rutas para cada combinación y se crean viajes a partir de combinaciones viables.
5) **Asignación:** Los viajes se ofrecen a conductores activos.



Sobre los estados:

Historia BK002-01: Agrupación de Solicitudes por PMCP y Tipo (Origen/Destino)

- **Estado Inicial:**
  - **Tabla:** solicitudes_viaje
  - **Estado:** pendiente
  - **Razón:** La historia comienza trabajando con solicitudes de viaje que aún no han sido procesadas ni asignadas, lo que corresponde al estado inicial pendiente.
- **Estados de Transición:**
  - **Tabla:** solicitudes_viaje

        - **Estado:** agrupada
        - **Razón:** Una vez que las solicitudes se asignan a un grupo de solicitudes candidatas, su estado cambia a agrupada, indicando que han sido procesadas por esta historia.
    - **Tabla:** grupos_solicitudes_candidatos
        - **Estado:** nuevo_grupo
        - **Razón:** Al crear un nuevo grupo de solicitudes candidatas, se insertan registros en esta tabla con el estado inicial nuevo_grupo.

Historia BK002-02: Generación de Combinaciones de Pasajeros para Posibles Viajes

- **Estado Inicial:**
    - **Tabla:** grupos_solicitudes_candidatos
    - **Estado:** nuevo_grupo
    - **Razón:** La historia inicia con grupos recién creados por BK002-01, que están en estado nuevo_grupo, listos para ser procesados y generar combinaciones.
- **Estados de Transición:**
    - **Tabla:** grupos_solicitudes_candidatos
        - **Estado:** combinaciones_generadas
        - **Razón:** Una vez generadas las combinaciones de pasajeros, el estado del grupo se actualiza a combinaciones_generadas, indicando que el procesamiento ha avanzado.
    - **Tabla:** combinaciones_viaje_propuestas
        - **Estado:** optimizacion_pendiente
        - **Razón:** Se crean registros para cada combinación propuesta, con un estado inicial de optimizacion_pendiente, ya que estas combinaciones esperan ser optimizadas en la siguiente historia.

Historia BK002-03: Optimización de Rutas y Cálculo de Métricas para Posibles Viajes

- **Estado Inicial:**
    - **Tabla:** combinaciones_viaje_propuestas
    - **Estado:** optimizacion_pendiente
    - **Razón:** La historia trabaja con combinaciones generadas por BK002-02 que están pendientes de optimización, correspondientes al estado optimizacion_pendiente.
- **Estados de Transición:**
    - **Tabla:** combinaciones_viaje_propuestas
        - **Estados:** optimizada_oferta_creada, descartada_no_rentable, o descartada_ruta_inviable
        - **Razón:** Dependiendo del resultado de la optimización, las combinaciones pueden transicionar a optimizada_oferta_creada (si se genera una oferta viable), o a estados de descarte como descartada_no_rentable (si no es rentable) o descartada_ruta_inviable (si la ruta no es factible).

- **Tabla:** viajes
    - **Estado:** disponible
    - **Razón:** Si la optimización es exitosa y se crea una oferta, se insertan registros en la tabla viajes con el estado inicial disponible, listos para ser ofrecidos a conductores.

Historia BK003: Endpoint para Listar Ofertas de Viaje Cercanas a Conductores

Descripción: Esta historia proporciona un endpoint que permite a los conductores ver ofertas de viaje cercanas a su ubicación actual.

- **Estado Inicial:**
    - **Tabla:** viajes
    - **Estado:** disponible
    - **Razón:** El endpoint consulta viajes que están en estado disponible y que no tienen un conductor asignado (conductor_id IS NULL), ya que estos son los viajes ofertados a conductores.
- **Estados de Transición:**
    - **Tabla:** viajes
        - **Estado:** No hay transición directa en esta historia; sin embargo, podría implicar aceptado_conductor en un proceso posterior.
        - **Razón:** BK003 se centra en listar ofertas y no modifica estados directamente. Si un conductor acepta una oferta, el estado del viaje cambiaría a aceptado_conductor, pero esto se consideraría parte de otra historia o acción fuera del alcance inmediato de BK003.

Ejecución Periódica (Cron Jobs):

bk002_01_groupRequests.js: Deberia ejecutarse cada X minutos (ej. cada 1-5 minutos) para procesar nuevas solicitudes_viaje pendientes.

bk002_02_generateCombinations.js: Podría ejecutarse después de BK002-01 o también periódicamente, buscando grupos_solicitudes_candidatos con estado 'nuevo_grupo' o 'combinaciones_pendientes'.

bk002_03_optimizeTrips.js: Similarmente, se ejecutaría buscando combinaciones_viaje_propuestas con estado 'optimizacion_pendiente'.

Implementación: librería node-cron dentro del archivo cronJobs.js dedicado para programar la ejecución de estas funciones.