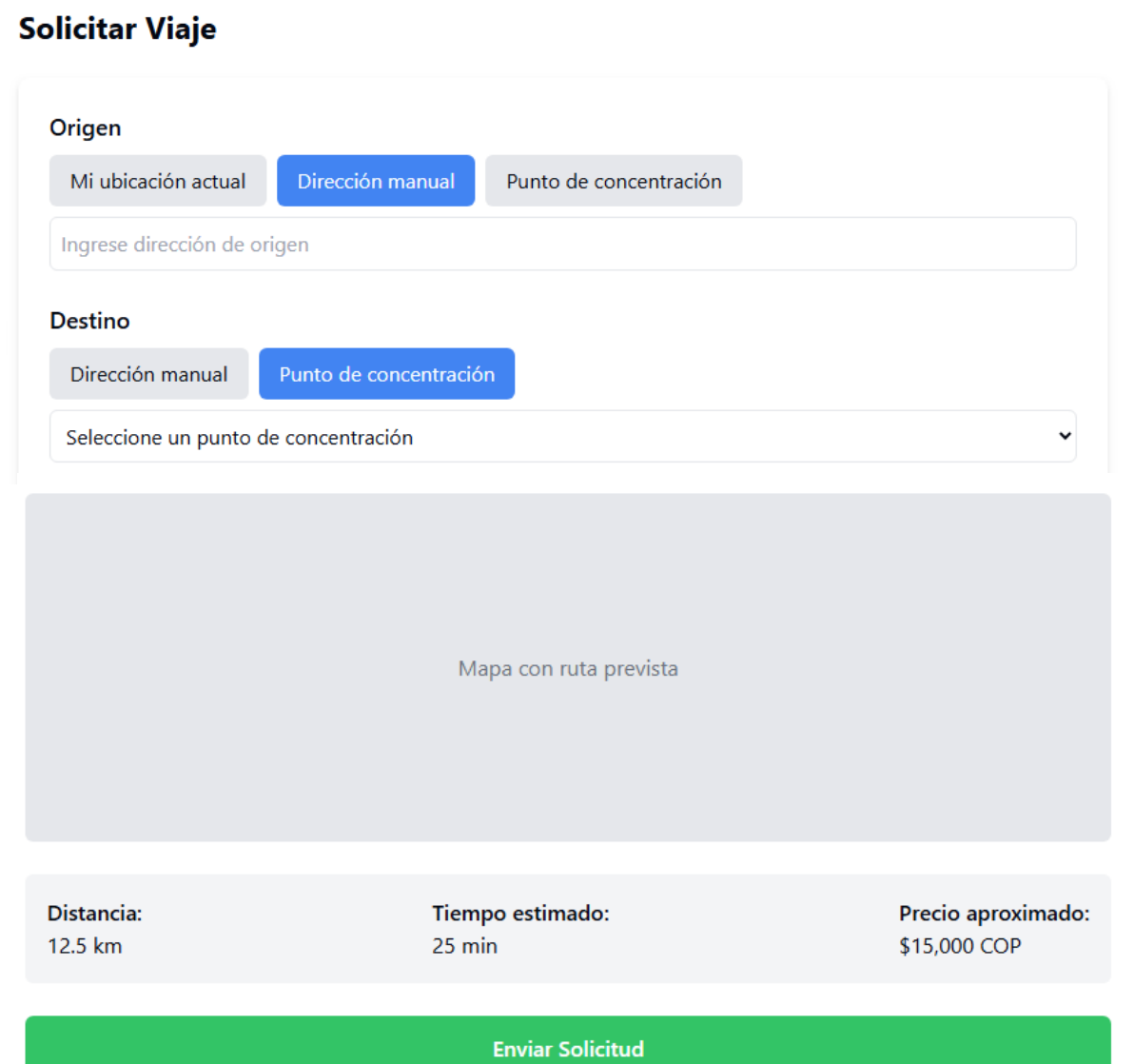
MODULO TRANSPORTE COMPARTIDO DINAMICO

Se define un módulo de transporte dinámico donde pasajeros solicitan viajes indicando origen y destino (al menos uno debe ser un “punto de alta concentración”), y conductores eligen paquetes de viajes ya agrupados por cercanía y rentabilidad. El sistema geolocaliza al usuario (HTML5 Geolocation), distingue si está en un punto predefinido o particular, muestra mapa con rutas y costos estimados, permite cancelar antes de asignación, notifica cuando un conductor acepta y despliega detalles. Para los conductores, se generan “ofertas de viaje” basadas en subgrupos de pasajeros optimizados de acuerdo a la capacidad de los vehículos de los conductores disponibles en un momento dado (ofreciendo servicios de transporte compartido); cada oferta incluye puntos de recolección/desembarque, tiempos, distancias y ganancias estimada.

FRONT-END

HU007-Solicitar-viaje (Vista pasajeros)



US-P01: Geolocalización Automática

La aplicación web debe detectar la ubicación actual del pasajero (HTML5 Geolocation + reverse-geocoding gratuito). De manera que si el pasajero está en un punto de alta concentración o punto particular sea detectado automáticamente *(esto comprobación debería realizarse desde el backend)*.

Criterios de Aceptación

- Usa la API de Geolocation del navegador
- Si el pasajero en **radio <100 m** de un **"punto de alta concentración"**, se marca automáticamente
- Si no, se le invita al pasajero a confirmar o teclear dirección

US-P02: Selección de Destino

El pasajero puede seleccionar destino ya sea un punto de alta concentración (lista desplegable) o escribir dirección libre

Criterios de Aceptación

Lista con funcionalidad de búsqueda con los puntos de alta concentración de personas definidos.

Validación de mínimo un punto de alta concentración en origen o destino

HU008-Activar-servicios (Vista conductores)

Objetivo: La idea es saber que conductores estan activos en un momento dado y dispónibles para realizar viajes (Ya que tenemos una lista de conductores, pero puede que muchos de estos conductores no esten activos en la plataforma, por lo que, es de vital importancia saber que conductores están activos y disponibles para tenerlos en encuenta en el empajeramiento de viajes)

Implementación:

- **Base de datos:**

Se ha creado una nueva tabla con las siguientes columnas:

```
TABLE public.conductores_activos_disponibles (
    conductor_id    INT    NOT    NULL    PRIMARY    KEY    REFERENCES
public.conductores(id) ON DELETE CASCADE,
    ubicacion_actual_lat DECIMAL(9,6) NOT NULL,
    ubicacion_actual_lon DECIMAL(9,6) NOT NULL,
    pmcp_cercano_id   INT   REFERENCES   public.puntos_concentracion(id)
DEFAULT NULL
    ultima_actualizacion_ubicacion TIMESTAMP WITHOUT TIME ZONE NOT NULL
DEFAULT CURRENT_TIMESTAMP
    sesion_expira_en TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    estado_disponibilidad_viaje    VARCHAR(30)    NOT    NULL    DEFAULT
'disponible'   CHECK   (estado_disponibilidad_viaje   IN   ('disponible',
'ofrecido_viaje', 'en_viaje_asignado')),
    created_at TIMESTAMP WITHOUT TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITHOUT TIME ZONE DEFAULT CURRENT_TIMESTAMP
```

```
);
```

- **En el frontend:**

Cuando un conductor accede a la pagina de “Buscar viaje” y dicho conductor no se encuentra en la tabla “conductores_activos_disponibles” se debe mostrar a pantalla completa el siguiente mensaje “¿Estas interesado en ofrecer servicios de transporte compartido en estos momentos?”

“Activa tus servicios, botón: Haz clic aquí”

- **En el backend:**

Cuando el conductor hace clic en el botón “Haz clic aquí” el Backend hace un INSERT INTO conductores_activos_disponibles, y agrega el conductor, de esta manera se sabe que esta activo.

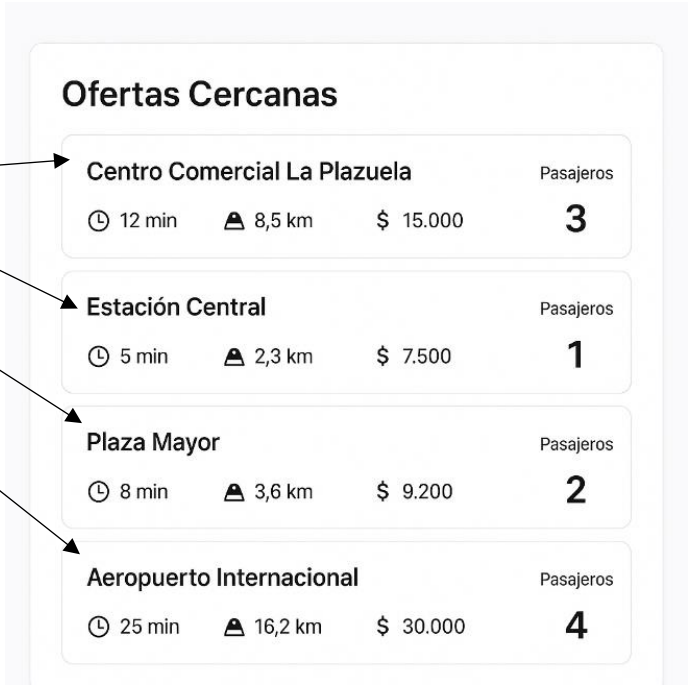
Se hará uso de dos implementaciones importantes:

- **Desactivación** (Automática/Cron Job): Un proceso backend ejecuta periódicamente: DELETE FROM conductores_activos_disponibles WHERE sesion_expira_en < NOW();. (Esto es para evitar que el estado de los conductores quede activo indefinidamente, la idea es que después de 1 hora, se elimine el estado de disponibilidad de los conductores (es decir, eliminado de la tabla “conductores_activos_disponibles” dicho registro de conductor.
- **Heartbeat**: Cada ping actualiza ubicacion_actual_lat/lon y ultima_actualizacion_ubicacion = NOW().

HU009-Lista-viajes (Vista conductores)

El título a mostrar es el punto de mayor concentración de personas.

Todo viaje siempre alguno de sus puntos, sea destino o origen, será un PMCP



US-C01: Lista de viajes

Los conductores pueden ver una lista de viajes creados automáticamente desde el back-end según su cercanía a un punto origen/destino de algún pasajero o de un punto de mayor concentración de personas.

Los conductores pueden elegir que viaje tomar, pero una vez que toma un viaje, dicho viaje desaparece de la lista y debe aparecer en la sección de "Mis viajes" de la vista conductores.

[UI]-botón: "Tomar viaje"

[UI]-botón: "Ver detalles del viaje"

Criterios de Aceptación

- Cada oferta muestra: punto de alta concentración, estimado de tiempo, km, ganancia y nº de pasajeros
- Ordenadas por proximidad al conductor

HU0010-Detalles-viaje (Vista conductores)

US-C02: Ver Detalle de Viaje

Cuando el conductor hacer clic sobre el botón "Ver detalles del viaje" se debe poder ver mapa y lista de ubicaciones de recogida y desembarque de cada pasajero

Criterios de Aceptación

- Se abre panel lateral con datos y mapa interactivo (Leaflet)

BACK-END

BK002-Funciones-Emparejamiento-ubicaciones

Viajes rentables

Diseñar e implementar el algoritmo que agrupa solicitudes de viaje en paquetes de pasajeros (el rango de tamaños de estos "paquetes" dependería de la capacidad de los vehículos de los conductores disponibles en un momento dado) de modo que:

1) Uno de sus puntos (origen o destino) sea un PMCP (punto de mayor concentración).
2) La ruta total sea la más corta posible (distancia mínima).
3) Se garantice la rentabilidad del viaje para el conductor (umbral mínimo de ganancia).
4) Se maximice la ocupación del vehículo.

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

Trabajar en "workers/bk002_01_groupRequests.js"

Objetivo: Diseñar una función/servicio para procesar las solicitudes de viaje pendientes de la tabla solicitudes_viaje y agruparlas según el PMCP que tienen en común, distinguiendo si el PMCP es el origen o el destino.

Accesos:

- Acceso a la tabla solicitudes_viaje (filtrando por estado = 'pendiente').
- Acceso a la tabla puntos_concentracion.

Flujo de trabajo

1. Lee solicitudes_viaje con estado 'pendiente'.
2. Crea registros en grupos_solicitudes_candidatos y solicitudes_en_grupo_candidato.
3. Actualiza grupos_solicitudes_candidatos.estado_procesamiento a 'nuevo'.
4. Podrías actualizar solicitudes_viaje.estado a 'en_agrupacion'.

Input

```
[
  {
    "id": 1,
```

```
    "pasajero_id": 101,
    "origen_lat": 4.5,
    "origen_lon": -74.1,
    "destino_lat": 4.8,
    "destino_lon": -74.3,
    "es_origen_concentracion": true,
    "es_destino_concentracion": false,
    "pmcp_id_origen": 1,
    "pmcp_id_destino": null
  },
  {
    "id": 2,
    "pasajero_id": 102,
    "origen_lat": 4.5,
    "origen_lon": -74.1,
    "destino_lat": 4.9,
    "destino_lon": -74.4,
    "es_origen_concentracion": true,
    "es_destino_concentracion": false,
    "pmcp_id_origen": 1,
    "pmcp_id_destino": null
  },
  {
    "id": 3,
    "pasajero_id": 103,
    "origen_lat": 4.6,
    "origen_lon": -74.2,
    "destino_lat": 4.5,
    "destino_lon": -74.1,
    "es_origen_concentracion": false,
    "es_destino_concentracion": true,
    "pmcp_id_origen": null,
    "pmcp_id_destino": 1
  },
  {
    "id": 4,
    "pasajero_id": 104,
    "origen_lat": 6.2,
    "origen_lon": -75.5,
    "destino_lat": 6.0,
    "destino_lon": -75.3,
    "es_origen_concentracion": true,
    "es_destino_concentracion": false,
    "pmcp_id_origen": 2,
    "pmcp_id_destino": null
  }
]
```


Output guardar en table “grupos_solicitudes_candidatos”


```
{
  "solicitudes_agrupadas_por_pmcp": [
    {
```

```
      "pmcp_id": 1,
      "pmcp_es_origen_del_grupo": true,
      "solicitudes": [
        {
          "solicitud_id": 1,
          "pasajero_id": 101,
          "origen_lat": 4.5,
          "origen_lon": -74.1,
          "destino_lat": 4.8,
          "destino_lon": -74.3
        },
        {
          "solicitud_id": 2,
          "pasajero_id": 102,
          "origen_lat": 4.5,
          "origen_lon": -74.1,
          "destino_lat": 4.9,
          "destino_lon": -74.4
        }
      ]
    },
    {
      "pmcp_id": 1,
      "pmcp_es_origen_del_grupo": false,
      "solicitudes": [
        {
          "solicitud_id": 3,
          "pasajero_id": 103,
          "origen_lat": 4.6,
          "origen_lon": -74.2,
          "destino_lat": 4.5,
          "destino_lon": -74.1
        }
      ]
    },
    {
      "pmcp_id": 2,
      "pmcp_es_origen_del_grupo": true,
      "solicitudes": [
        {
          "solicitud_id": 4,
          "pasajero_id": 104,
          "origen_lat": 6.2,
          "origen_lon": -75.5,
          "destino_lat": 6.0,
          "destino_lon": -75.3
        }
      ]
    }
  ]
}
```


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

Trabajar en “bk002_02_generateCombinations.js”

Objetivo: El output de la historia “**BK002-01”** es el input de esta historia. El objetivo es diseñar una función/servicio para generar todas las combinaciones posibles de pasajeros que podrían formar un viaje, respetando la capacidad de los vehículos.

Tareas Técnicas:

Para cada grupo de solicitudes_agrupadas_por_pmcp:

- Si el número de solicitudes en el grupo es menor a alguna capacidad de algún vehículo dentro de los **conductores_activos_disponibles**, ignorar este grupo para combinaciones.
- Generar todas las combinaciones de k pasajeros, donde k está entre el mínimo y máximo de la capacidad de vehículos de los **conductores_activos_disponibles**.
- Cada combinación representa un "posible viaje" preliminar.

Flujo de trabajo:

1. Lee grupos_solicitudes_candidatos con estado 'nuevo' (o similar).
2. Para cada grupo, genera combinaciones y las inserta en combinaciones_viaje_propuestas y solicitudes_en_combinacion_propuesta.
3. Actualiza combinaciones_viaje_propuestas.estado_procesamiento a 'optimizacion_pendiente'.
4. Actualiza grupos_solicitudes_candidatos.estado_procesamiento a 'combinaciones_generadas'.

Input1 (Output de BK002-01)


```
{
  "solicitudes_agrupadas_por_pmcp": [
    {
      "pmcp_id": 1,
      "pmcp_es_origen_del_grupo": true,
      "solicitudes": [
        {
          "solicitud_id": 1,
          "pasajero_id": 101,
          "origen_lat": 4.5,
          "origen_lon": -74.1,
          "destino_lat": 4.8,
          "destino_lon": -74.3
        },
        {
          "solicitud_id": 2,
          "pasajero_id": 102,
          "origen_lat": 4.5,
          "origen_lon": -74.1,
          "destino_lat": 4.9,
          "destino_lon": -74.4
        }
```

```
      ]
    },
    {
      "pmcp_id": 1,
      "pmcp_es_origen_del_grupo": false,
      "solicitudes": [
        {
          "solicitud_id": 3,
          "pasajero_id": 103,
          "origen_lat": 4.6,
          "origen_lon": -74.2,
          "destino_lat": 4.5,
          "destino_lon": -74.1
        }
      ]
    },
    {
      "pmcp_id": 2,
      "pmcp_es_origen_del_grupo": true,
      "solicitudes": [
        {
          "solicitud_id": 4,
          "pasajero_id": 104,
          "origen_lat": 6.2,
          "origen_lon": -75.5,
          "destino_lat": 6.0,
          "destino_lon": -75.3
        }
      ]
    }
  ]
}
```

Input2

```
{
  "conductores_activos_disponibles": [
    {
      "id_conductor": 201,
      "ubicacion_latitud": 4.6,
      "ubicacion_longitud": -74.2,
      "vehiculo_capacidad": 4,
      "esta_en_un_PMCP": true
    },
    {
      "id_conductor": 202,
      "ubicacion_latitud": 6.3,
      "ubicacion_longitud": -75.4,
      "vehiculo_capacidad": 3,
      "esta_en_un_PMCP": false
    }
    // más conductores activos
  ]
}
```

Output guardar en la tabla “combinaciones_viaje_propuestas”


```
{
  "posibles_viajes_combinaciones": [
    {
      "pmcp_id": 1,
      "pmcp_es_origen_del_grupo": true,
      "combinaciones": [
        {
          "pasajeros_participantes": [
            // Combinación de 3 pasajeros
            {
              "solicitud_id": 1,
              "pasajero_id": 101,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.8,
              "destino_lon": -74.3
            },
            {
              "solicitud_id": 2,
              "pasajero_id": 102,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.9,
              "destino_lon": -74.4
            },
            {
              "solicitud_id": 5,
              "pasajero_id": 105,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.7,
              "destino_lon": -74.2
            }
          ],
          "capacidad_utilizada": 3
        },
        {
          "pasajeros_participantes": [
            // Otra combinación de 3
            {
              "solicitud_id": 1,
              "pasajero_id": 101,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.8,
              "destino_lon": -74.3
            },
            {
              "solicitud_id": 2,
              "pasajero_id": 102,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.9,
              "destino_lon": -74.4
```

```
            },
            {
              "solicitud_id": 6,
              "pasajero_id": 106,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.6,
              "destino_lon": -74.0
            }
          ],
          "capacidad_utilizada": 3
        },
        {
          "pasajeros_participantes": [
            // Combinación de 4 pasajeros
            {
              "solicitud_id": 1,
              "pasajero_id": 101,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.8,
              "destino_lon": -74.3
            },
            {
              "solicitud_id": 2,
              "pasajero_id": 102,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.9,
              "destino_lon": -74.4
            },
            {
              "solicitud_id": 5,
              "pasajero_id": 105,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.7,
              "destino_lon": -74.2
            },
            {
              "solicitud_id": 6,
              "pasajero_id": 106,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.6,
              "destino_lon": -74.0
            }
          ],
          "capacidad_utilizada": 4
        }
        // ... más combinaciones
      ]
    }
    // ... combinaciones para otros grupos de PMCP
  ]
```

```
}
```

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

Trabajar en "bk002_03_optimizeTrips.js"

Objetivo: Para cada "posible viaje" *(combinación de pasajeros - Ouput de la historia BK002-02)*. Se necesita determinar la secuencia óptima de paradas (recogidas/entregas), calcular la distancia total, el tiempo estimado y la ganancia estimada. La ruta debe ser la más corta posible minimizando desvíos.

Tareas Técnicas:

Para cada combinación de posibles_viajes_combinaciones:

1. Obtener las coordenadas del PMCP involucrado desde la tabla puntos_concentracion usando pmcp_id.
2. Construir la lista de waypoints:
    - Si pmcp_es_origen_del_grupo: El PMCP es el punto de inicio. Los siguientes waypoints son los destinos particulares de cada pasajero en la combinación.
    - Si pmcp_es_origen_del_grupo es false: Los waypoints iniciales son los orígenes particulares de cada pasajero. El PMCP es el punto final común.
3. **Optimización de Secuencia de Paradas (Problema del Viajante - TSP):**
    - Si pmcp_es_origen_del_grupo: El PMCP es fijo como inicio. Se necesita encontrar el orden óptimo para visitar los N destinos.
    - Si pmcp_es_origen_del_grupo es false: Se necesita encontrar el orden óptimo para visitar los N orígenes. El PMCP es fijo como final.
    - Para N entre (la capacidad mínima y máxima de los vehiculo de conductores disponibles), se pueden explorar algoritmos exactos (si se usa una matriz de distancias precalculada entre todos los puntos) o heurísticas eficientes (Nearest Neighbor, algoritmos genéticos simples).
    - Utilizar un servicio de ruteo (ej. OSRM auto-hosteado, o API pública de OpenRouteService/GraphHopper con límites de uso gratuito) para obtener la matriz de distancias/tiempos entre todos los puntos relevantes (PMCP y orígenes/destinos particulares de la combinación).
4. Una vez obtenida la secuencia óptima de paradas, usar el servicio de ruteo nuevamente para obtener la ruta completa (GeoJSON), la distancia total y el tiempo estimado para esa secuencia.
5. Calcular la ganancia estimada: Ganancia = (Suma de tarifas de pasajeros en la combinación) - (Costo operativo del viaje). El

costo operativo podría ser un % de la tarifa o basado en distancia/tiempo. Se necesita definir una política de tarifas por pasajero (por distancia).

Flujo de trabajo

1. Lee combinaciones_viaje_propuestas con estado 'optimizacion_pendiente'.
2. Realiza la optimización de ruta y cálculo de métricas.
3. Si la combinación es viable y cumple criterios (rentabilidad, etc.):
4. Crea un nuevo registro en la tabla viajes (con estado = 'disponible', conductor_id = NULL).
5. Crea los registros correspondientes en viaje_pasajeros (usando solicitud_viaje_id de solicitudes_en_combinacion_propuesta).
6. Actualiza el estado de las solicitudes_viaje originales a 'ofertado' (o similar, indicando que ahora forman parte de un viaje potencial).
7. Actualiza combinaciones_viaje_propuestas.estado_procesamiento a 'optimizada_oferta_creada'.
8. Si no es viable, actualiza combinaciones_viaje_propuestas.estado_procesamiento a 'descartada_...'

Input (Output BK002-02)

```
{
  "posibles_viajes_combinaciones": [
    {
      "pmcp_id": 1,
      "pmcp_es_origen_del_grupo": true,
      "combinaciones": [
        {
          "pasajeros_participantes": [
            // Combinación de 3 pasajeros
            {
              "solicitud_id": 1,
              "pasajero_id": 101,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.8,
              "destino_lon": -74.3
            },
            {
              "solicitud_id": 2,
              "pasajero_id": 102,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.9,
              "destino_lon": -74.4
            },
            {
              "solicitud_id": 5,
              "pasajero_id": 105,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
```

```
            "destino_lat": 4.7,
            "destino_lon": -74.2
          }
        ],
        "capacidad_utilizada": 3
      },
      {
        "pasajeros_participantes": [
          // Otra combinación de 3
          {
            "solicitud_id": 1,
            "pasajero_id": 101,
            "origen_lat": 4.5,
            "origen_lon": -74.1,
            "destino_lat": 4.8,
            "destino_lon": -74.3
          },
          {
            "solicitud_id": 2,
            "pasajero_id": 102,
            "origen_lat": 4.5,
            "origen_lon": -74.1,
            "destino_lat": 4.9,
            "destino_lon": -74.4
          },
          {
            "solicitud_id": 6,
            "pasajero_id": 106,
            "origen_lat": 4.5,
            "origen_lon": -74.1,
            "destino_lat": 4.6,
            "destino_lon": -74.0
          }
        ],
        "capacidad_utilizada": 3
      },
      {
        "pasajeros_participantes": [
          // Combinación de 4 pasajeros
          {
            "solicitud_id": 1,
            "pasajero_id": 101,
            "origen_lat": 4.5,
            "origen_lon": -74.1,
            "destino_lat": 4.8,
            "destino_lon": -74.3
          },
          {
            "solicitud_id": 2,
            "pasajero_id": 102,
            "origen_lat": 4.5,
            "origen_lon": -74.1,
            "destino_lat": 4.9,
            "destino_lon": -74.4
          },
```

```
            {
              "solicitud_id": 5,
              "pasajero_id": 105,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.7,
              "destino_lon": -74.2
            },
            {
              "solicitud_id": 6,
              "pasajero_id": 106,
              "origen_lat": 4.5,
              "origen_lon": -74.1,
              "destino_lat": 4.6,
              "destino_lon": -74.0
            }
          ],
          "capacidad_utilizada": 4
        }
        // ... más combinaciones
      ]
    }
    // ... combinaciones para otros grupos de PMCP
  ]
}
```

Output

```
{
  "viajes_potenciales_optimizados": [
    {
      "pmcp_id": 1,
      "pmcp_es_origen_del_grupo": true,
      "pasajeros_participantes": [
        // IDs de las solicitudes originales o pasajeros_id
        {
          "solicitud_id": 1,
          "pasajero_id": 101
        },
        {
          "solicitud_id": 2,
          "pasajero_id": 102
        },
        {
          "solicitud_id": 5,
          "pasajero_id": 105
        }
      ],
      "capacidad_utilizada": 3,
      "orden_paradas_optimizado": [
        // Objetos con tipo (recogida/entrega) y coords, o IDs
        // Ejemplo si PMCP es origen:
        {
          "tipo": "pmcp_origen",
          "lat": 4.5,
```

```
          "lon": -74.1,
          "pmcp_id": 1
        },
        {
          "tipo": "entrega",
          "pasajero_id": 105,
          "lat": 4.7,
          "lon": -74.2
        }, // Supongamos D3 es el más cercano
        {
          "tipo": "entrega",
          "pasajero_id": 101,
          "lat": 4.8,
          "lon": -74.3
        }, // Luego D1
        {
          "tipo": "entrega",
          "pasajero_id": 102,
          "lat": 4.9,
          "lon": -74.4
        }  // Finalmente D2
      ],
      "ruta_geojson": {
        "type": "LineString",
        "coordinates": [
          [-74.1, 4.5],
          [-74.2, 4.7]
          // ... más coordenadas
        ]
      },
      "distancia_km": 12.5,
      "tiempo_estimado_min": 25,
      "ganancia_estimada_cop":  15000  //  (Ej:  3  pasajeros  *  7000
tarifa_prom - 6000 costo_op)
    }
    // ... más viajes potenciales optimizados
  ]
}
```

BK003- Endpoint para Listar Ofertas de Viaje Cercanas a Conductores

Cuando el conductor accede a la vista HU009-Lista-viajes de la vista conductores, el conductor debe ver una lista de "Ofertas Cercanas" (viajes disponibles creados por **BK002-03**), ordenadas por proximidad a mi ubicación actual o al PMCP de origen del viaje.

Tareas Técnicas:

- Crear un endpoint API (GET /api/conductores/ofertas-viaje).
- El endpoint recibirá la ubicación actual del conductor (latitud, longitud) como parámetros de query.

- Consultar la tabla viajes por estado = 'disponible' y conductor_id IS NULL *(Si el conductor es null, es porque todavía el viaje no se ha asignado a un conductor)*.
- Para cada viaje disponible:
    1. Obtener las coordenadas del punto_concentracion_id asociado al viaje (este es el PMCP ancla del viaje).
    2. Si el PMCP es el origen del viaje, calcular la distancia entre la ubicación del conductor y este PMCP.
    3. Si el PMCP es el destino del viaje, la "cercanía" es más compleja. Podría ser la distancia al primer punto de recogida de ese viaje (requeriría consultar viaje_pasajeros y luego solicitudes_viaje para las coordenadas del primer pasajero en orden_recogida, **esto se habla mas adelante**).
- Ordenar los viajes por esta distancia calculada (menor distancia primero).

Input (Request)

```
GET /api/conductores/ofertas-viaje?lat=4.55&lon=-74.15
```

Output

```
{
  "ofertas_cercanas": [
    {
      "viaje_id": 501,
      "pmcp_info": {
        // Información del PMCP principal del viaje
        "nombre": "Centro Comercial La Plazuela",
        // De tabla puntos_concentracion
        "id": 1
      },
      // Distancia del conductor al PMCP (si es origen) o primer punto
de recogida.
      "distancia_conductor_a_inicio_viaje_km": 3.5,
      // De tabla 'viajes'
      "tiempo_total_viaje_estimado_min": 35,
      // De tabla 'viajes'
      "ganancia_estimada_cop": 20000,
      // De tabla 'viajes' (capacidad)
      "numero_pasajeros": 4
    },
    {
      "viaje_id": 502,
      "pmcp_info": {
        "nombre": "Estación Central",
        "id": 3
      },
      "distancia_conductor_a_inicio_viaje_km": 5.2,
      "tiempo_total_viaje_estimado_min": 20,
```

```
      "ganancia_estimada_cop": 12000,
      "numero_pasajeros": 3
    }
    // ...
  ]
}
```

BK003-Visualizacion-ruta-real

Se debe ir investigando como graficar una ruta que contiene tiene puntos de ubicaciones con componentes de latitud y longitud. El mapa debe mostrar los puntos importantes, y la línea de ruta que conecte los puntos de origen(es) con destino(s).