SPRINT 2

HU005-01-lista-conductores: Modulo transporte programado (Vista Pasajero)

- **Historia de Usuario:** Como **Pasajero con sesión iniciada**, quiero **ver una lista de conductores disponibles que ofrecen servicios de transporte programado** para que pueda **explorar sus perfiles y elegir uno para contactar.**
- Esta vista actúa como un directorio o catálogo de conductores que participan en el sistema de "Transporte Compartido Programado". Debe presentar información clave de forma concisa para permitir a los pasajeros comparar y seleccionar conductores.
- **Criterios de Aceptación:**
    1. **Obtención y Muestra de Datos:**
        - [FUNC] Al cargar la página, obtener una lista de conductores *activos*des de la API backend (GET /api/conductores).
        - [UI] Mostrar un indicador de carga mientras se obtienen los datos.
        - [UI] Si no se encuentran conductores, mostrar un mensaje amigable (p.ej., "No hay conductores disponibles para transporte programado en este momento.").
    2. **Lista/Cuadrícula de Conductores:**
        - [UI] Mostrar los conductores en formato de lista de tarjetas (Una sola columna, varias filas, ocupando gran parte del ancho.
        - [UI] Cada entrada de conductor (tarjeta/fila) debe mostrar:
            - **Foto de Perfil**
            - **Nombre Completo**
            - **Calificación Promedio:** Mostrada visualmente usando estrellas (p.ej., 4.5 estrellas de 5).
            - **Capacidad del Vehículo:** Texto como "Capacidad: X pasajeros" (de vehicles.capacity).
            - **Info Vehículo** vehicles.make y vehicles.model.
            - **Fecha de Ingreso:** Texto como "Miembro desde: Mes Año" (de tabla conductores.created_at).
        - [UI] Incluir un botón/enlace claro en cada entrada: "Ver Detalles del conductor y el botón de Contactar".
    3. **Filtrado/Ordenación:**

        - [UI] Añadir controles para filtrar por capacidad del vehículo u ordenar por calificación del conductor, o por id del conductor o nombre del conductor.

- **Logica Backend:**
    1. **Endpoint:** GET /api/conductores
    2. **Query Params (Opcional para filtros):** ?sortBy=rating&minCapacity=3&page=1&limit=10
        - Consultar la tabla conductor, haciendo join con vehiculos.
        - Aplicar filtrado/ordenación basado en query params.
        - Implementar paginación.
        - Devolver 200 OK con un array de objetos de conductor conteniendo los campos necesarios para el frontend (incluyendo la calificación promedio calculada/almacenada).
- // GET /api/conductores -> Ejemplo de Respuesta
- [
- {
- "driverId": 5,
- "firstName": "Ana",
- "lastName": "Garcia",
- "profilePictureUrl": "/uploads/drivers/ana_garcia.jpg",
- "averageRating": 4.8, // Calculada o almacenada
- "vehicle": {
- "make": "Chevrolet",
- "model": "Onix",
- "capacity": 4
- },
- "joinedDate": "2024-01-15T10:00:00Z"
- },
- // ... más conductores
-]

HU005-02-detalles-conductor: Ver Detalles del Conductor y Reseñas (Vista Pasajero)

- **Historia de Usuario:** Como **Pasajero con sesión iniciada explorando conductores**, quiero **hacer clic en un conductor para ver información más detallada, incluyendo reseñas de otros pasajeros, y poder añadir mi propia reseña** para que pueda **tomar una decisión informada antes de contactarlo y contribuir con retroalimentación a la comunidad.**
- **Criterios de Aceptación:**
    1. **Disparador de Visualización y Diseño:**
        - [UI] Cuando se selecciona un conductor de la lista (HU005-01), esta vista de detalle se hace visible.
        - [UI] Un patrón común es un diseño de dos columnas: la lista de conductores (potencialmente reducida) a la izquierda, y los detalles del conductor seleccionado ocupando la porción derecha de la pantalla.
        - [UI] Incluir una forma de cerrar/descartar la vista de detalle (p.ej., un botón 'X', hacer clic fuera del panel).
    2. **Información Detallada del Conductor:**
        - [FUNC] Obtener datos detallados para el ID de conductor *seleccionado* desde GET /api/drivers/conductores/:id_conductor.
        - [UI] Mostrar una versión más grande de la profile_picture_url.
        - [UI] Mostrar prominentemente first_name y last_name.
        - [UI] Reiterar Calificación Promedio (estrellas y número).
        - [UI] Mostrar joinedDate ("Miembro desde...").
        - [UI] Mostrar detalles completos de vehicles (Marca, Modelo, Año, Color, Capacidad).
        - [UI] Opcionalmente mostrar número de viajes completados
        - [UI] **Botón Contactar:** Un botón primario "Contactar a [Nombre del Conductor]" debe estar visible. Hacer clic aquí debería probablemente iniciar la interfaz de chat (parte de HU006, pero la *iniciación* ocurre aquí creando una conversación si no existe).
    3. **Sección de Reseñas:**
        - [UI] Etiquetar claramente la sección: "Comentarios y Calificaciones".
        - [FUNC] Obtener reseñas para este conductor específico desde la API (GET /api/conductores/:idconductor/reseñas
        - [UI] Mostrar reseñas en orden cronológico (más nuevas primero).

- [UI] Cada reseña debe mostrar:
    - Nombre del Reseñador (p.ej., "María P." - quizás solo nombre e inicial del apellido por privacidad).
    - Calificación dada (estrellas visuales).
    - Texto del comentario.
    - Fecha de la reseña.
- [UI] Si no existen reseñas, mostrar "Este conductor aún no tiene comentarios.".
- [UI] Implementar paginación para reseñas si la lista puede volverse muy larga.

4. **Formulario Añadir Reseña:**
    - [UI] Mostrar una sección/formulario "Deja tu comentario".
    - [UI] Incluir una entrada de calificación por estrellas (permitir seleccionar 1-5 estrellas, requerido).
    - [UI] Incluir un área de texto para el comentario (requerido, validación de longitud mín/máx útil).
    - [UI] Un botón "Enviar Comentario", habilitado solo cuando se proporcionan calificación y comentario.
    - [FUNC] Al enviar, mandar los datos de la reseña al backend (POST /api/conductores/:idconductor/reseñas).
    - [FUNC] En éxito: Limpiar el formulario, mostrar un mensaje de éxito (p.ej., "Comentario enviado. ¡Gracias!"), y *añadir dinámicamente la nueva reseña al principio de la lista de reseñas mostrada* sin necesidad de refrescar toda la página.
5. **API Backend:**
    - **Obtener Detalles del Conductor (+ Reseñas Recomendado):**
        - **Endpoint:** GET /api/conductores/ idconductor
        - **Lógica Backend:** Obtener datos del conductor (de tabla conductores), datos del vehículo (de tabla vehiculos), y reseñas asociadas (de tabla reseñas, ordenadas por created_at DESC). Hacer Join reviews con pasajeros (o users) para obtener nombres de reseñadores. Agregar total_viajes. recuperar valor almacenado de promedio de calificacion. Devolver 200 OK con un objeto JSON anidado.
        - **Ejemplo Estructura Respuesta:**

- // GET /api/conductores /5
- {
- "driverId": 5,
- "firstName": "Ana", "lastName": "Garcia", /* ... otros campos conductor */
- "averageRating": 4.8,
- "joinedDate": "2024-01-15T10:00:00Z",
- "totalTrips": 52, // Ejemplo si se rastrea
- "vehicle": { /* ... detalles vehículo ... */ },
- "reviews": [
- { "reviewId": 101, "reviewerName": "Carlos R.", "rating": 5, "comment": "Excelente servicio, muy puntual!", "createdAt": "2024-06-10T15:30:00Z" },
- { "reviewId": 95, "reviewerName": "Maria L.", "rating": 4, "comment": "Buen viaje, conductor amable.", "createdAt": "2024-06-08T09:00:00Z" }
- // ... más reseñas (potencialmente paginadas)
-]
- }

HU006-01-mensajes-rutas-programadas: Modudo transporte programado (Vista Conductor)

- **Historia de Usuario:** Como **Conductor con sesión iniciada**, quiero **ver mensajes de pasajeros que me contactaron sobre viajes programados y poder responderles** para que pueda **coordinar detalles, confirmar disponibilidad y acordar el servicio programado.**
- Esta vista sirve como la bandeja de entrada del conductor para el módulo "Transporte Compartido Programado". Facilita la comunicación directa iniciada por pasajeros que encontraron al conductor vía HU005. Debería asemejarse a una interfaz básica de chat o mensajería.
- **Criterios de Aceptación:**
    1. **Diseño de Bandeja de Entrada:**
        - [UI] Típicamente un diseño de dos paneles.
        - **Panel Izquierdo (Lista de Conversaciones):**

        - [FUNC] Obtener la lista de conversaciones de las que el conductor forma parte desde GET /api/conversaciones/id_conductor.
        - [UI] Mostrar cada conversación como un ítem mostrando: Nombre del Pasajero (passenger.firstName + passenger.lastName), fragmento del último mensaje, timestamp del último mensaje.
        - [UI] Resaltar visualmente las conversaciones no leídas (p.ej., texto en negrita, insignia de conteo no leído).
        - [UI] Hacer clic en un ítem de conversación carga sus mensajes en el panel derecho y lo marca como seleccionado/activo.
        - [UI] Si no existen conversaciones, mostrar "No tienes mensajes de pasajeros por ahora.".
    - **Panel Derecho (Ventana de Chat):**
        - [UI] Muestra los mensajes para la conversación *actualmente seleccionada*. Mostrar nombre del pasajero arriba.
        - [FUNC] Obtener mensajes para el conversationId seleccionado desde GET /api/conversations/:conversationId/messages.
        - [UI] Los mensajes deben mostrarse cronológicamente (más antiguos arriba, más nuevos abajo).
        - [UI] Diferenciar visualmente entre mensajes enviados por el conductor y mensajes recibidos del pasajero (p.ej., alineación, color de fondo). Mostrar nombre/avatar del remitente
        - [UI] Un área de entrada abajo con un campo de texto para escribir mensajes y un botón "Enviar".
2. **Funcionalidad de Mensajes:**
    - [FUNC] Escribir en el campo de entrada y hacer clic en "Enviar" (o presionar Enter) envía el mensaje vía POST /api/conversations/:conversationId/messages.
    - [FUNC] El mensaje enviado debe aparecer inmediatamente en la ventana de chat (luego confirmada por backend). Limpiar el campo de entrada después de enviar.
    - [FUNC] Deshabilitar botón Enviar si el campo de entrada está vacío.
3. **Iniciar Conversaciones (Manejado en Otro Lugar):**
    - [CONTEXTO] Esta vista principalmente *gestiona* conversaciones existentes. La lógica de iniciación ocurre cuando un pasajero hace clic en "Contactar" en HU005-02. Esa acción debe disparar una llamada backend como POST

/api/conversations/initiate que verifica si existe una conversación entre el pasajero (del JWT) y el idconductor objetivo; si no, crea un nuevo registro de conversación y registros de participantes.

4. **Contratos API Backend:**
    - **Obtener Mis Conversaciones:**
        - **Endpoint:** GET /api/conversations/id
        - **Lógica Backend:** Obtener idconductor .Consultar tabla conversaciones, haciendo join para encontrar conversaciones que incluyen el idconductor. Marcar estado no leído basado en messages.read_status y messages.sender_user_id != driverId. Devolver 200 OK.
    - **Obtener Mensajes para una Conversación:**
        - **Endpoint:** GET /api/conversaciones/:idconversacion/mensajes
        - **Lógica Backend:** Verificar que el conductor (del JWT) sea participante en idconversacion. Obtener todos los mensajes para esta conversación, ordenar por created_at ASC. Hacer Join con users para obtener info del remitente si es necesario.
    - **Enviar Mensaje:**
        - **Endpoint:** POST /api/conversacion/:idconversacion/mensajes
        - **Cuerpo de la Solicitud (application/json):** {"messageText": "Hola, sí estoy disponible el viernes."}
        - **Lógica Backend:** Verificar que el conductor (del JWT) sea participante. Validar messageText. Insertar nuevo mensaje (idconveersacion, sender_user_id, message_text, read_status = false).
    - **Iniciar Conversación (Ejemplo - Llamado desde HU005-02):**
        - **Endpoint:** POST /api/conversacion/iniciar
        - **Cuerpo Solicitud:** {"targetDriverId": 5}
        - **Lógica Backend:** Obtener idpasajero. Verificar si ya existe una conversación entre idpsajero y idconductor. Si sí, devolver el idconversacion existente. Si no, crear un nuevo registro en conversaciones, crear dos registros en conversation_participants, y devolver el nuevo conversationId

HU006-02-mensajes-rutas-programadas: Modulo transporte programado (Vista Pasajero)

Al hacer clic sobre el botón “Contactar” de la lista de conductores, al usuario tipo pasajero se le desplegara un chat interactivo en donde el usuario podrá contactarse con el conductor, se debe aplicar la misma lógica de iniciación de conversaciones de la historia HU006-02