Proyecto de fin de asignatura — Reservas de Bus Intermunicipal y Encomiendas

1) Historia

En muchas rutas intermunicipales de LATAM los pasajes aún se venden en taquilla o WhatsApp, y las **encomiendas** (paquetes) viajan en el mismo bus. La empresa necesita un backend para **vender tiquetes con silla numerada**, gestionar **puntos de recogida/descenso**, admitir **pagos mixtos** (efectivo/transferencia/QR), y **rastrear encomiendas** con comprobantes de entrega, todo con **operación offline** en terminales con mala conexión.

---

2) Alcance funcional (MVP sólido)

- **Roles**: PASSENGER, CLERK (taquilla), DRIVER, DISPATCHER, ADMIN.
- **Catálogo de rutas** con horarios, paradas intermedias y flota (bus, placas, capacidad, comodidades).
- **Reserva de sillas** numeradas con selección en mapa y *overbooking* controlado.
- **Tarifas** por tramo y **precios dinámicos** por ocupación/hora pico.
- **Pagos**: QR/transferencia, **contraentrega** en taquilla o al subir; comprobante digital.
- **Encomiendas**: registro, etiquetado con código, trazabilidad de estados y entrega con OTP/foto.
- **Operación offline**: modo taquilla/driver que sincroniza al recuperar señal.
- **Notificaciones** (mock) por WhatsApp/SMS: compra, cambio de andén, llegada próxima.
- **Panel de despacho**: asignar bus a salida, abrir/cerrar abordaje, control de ocupación y equipaje.

---

3) Reglas de negocio clave

1. **Silla única**: una silla puede estar en **RESERVED** por 10 min (hold) y pasa a **SOLD** al pagar.

2. **Paradas intermedias**: sillas se liberan por tramos; permitir venta parcial Origen→Parada y Parada→Destino si no solapan.
3. **Menores y descuentos**: tarifas especiales (niño/estudiante/adulto mayor) con validaciones.
4. **Overbooking controlado**: porcentaje máximo (ej. 5%) por cancelaciones de última hora; requiere aprobación DISPATCHER.
5. **No-show**: si el pasajero no aborda antes de t-5 min, la silla vuelve a venta rápida; se cobra fee configurable.
6. **Equipaje**: límite de peso y cobro por exceso; etiqueta y conteo por maletero.
7. **Encomiendas**: requieren datos de remitente/destinatario; entrega con **OTP** y foto; si falla → INCIDENT.
8. **Seguridad**: autenticación del DRIVER en la salida; checklist del bus (SOAT, revisión) vigente.
9. **Cancelaciones**: con política según antelación (reembolso %).

---

4) Modelo de datos (resumen)

User(id, name, email, phone, role{PASSENGER|CLERK|DRIVER|DISPATCHER|ADMIN}, status, passwordHash, createdAt)

Route(id, code, name, origin, destination, distanceKm, durationMin)

Stop(id, routeId, name, order, lat, lng)

Bus(id, plate, capacity, amenities(json), status)

Trip(id, routeId, busId, date, departureAt, arrivalEta, status{SCHEDULED|BOARDING|DEPARTED|ARRIVED|CANCELLED})

Seat(id, busId, number, type{STANDARD|PREFERENTIAL})

FareRule(id, routeId, fromStopId, toStopId, basePrice, discounts(json), dynamicPricing{ON|OFF})

SeatHold(id, tripId, seatNumber, userId, expiresAt, status{HOLD|EXPIRED})

Ticket(id, tripId, passengerId, seatNumber, fromStopId, toStopId, price, paymentMethod{CASH|TRANSFER|QR|CARD}, status{SOLD|CANCELLED|NO_SHOW}, qrCode)

Baggage(id, ticketId, weightKg, fee, tagCode)

Parcel(id, code, senderName, senderPhone, receiverName, receiverPhone, fromStopId, toStopId, price, status{CREATED|IN_TRANSIT|DELIVERED|FAILED}, proofPhotoUrl, deliveryOtp)

Assignment(id, tripId, driverId, dispatcherId, checklistOk, assignedAt)

Incident(id, entityType{TRIP|TICKET|PARCEL}, entityId, type{SECURITY|DELIVERY_FAIL|OVERBOOK|VEHICLE}, note, createdAt)

Config(key, value)

---

5) Endpoints REST (ejemplos)

Auth & Users

- POST /api/auth/register (PASSENGER) / POST /api/auth/login — JWT

Catálogo

- GET /api/routes / GET /api/routes/{id}/stops
- GET /api/trips?routeId=&date= — salidas disponibles

Tickets & Seats

- POST /api/trips/{id}/seats/{seat}/hold — bloquear 10 min
- POST /api/trips/{id}/tickets — comprar (validación de tramo y precio)
- POST /api/tickets/{id}/cancel — política de reembolso

Despacho

- POST /api/trips/{id}/assign (DISPATCHER) — asignar bus/driver
- POST /api/trips/{id}/boarding/open|close — abrir/cerrar abordaje
- POST /api/trips/{id}/depart — salida; valida checklist

Encomiendas

- POST /api/parcels — crear envío
- POST /api/parcels/{code}/status — IN_TRANSIT|DELIVERED|FAILED (OTP/foto)

Pagos & Cierre

- POST /api/payments/confirm — QR/transferencia
- POST /api/cash/close (CLERK/DRIVER) — cierre de caja

Admin

- PUT /api/admin/config — tiempos de hold, fees, descuentos, % overbooking
- GET /api/admin/metrics — ocupación, cancelaciones, puntualidad, ingresos

---

6) Casos de uso (criterios de aceptación)

1. **Selección de tramo**: un pasajero compra Bogotá→Tunja en trip con paradas; el sistema solo bloquea el tramo correspondiente.
2. **Hold & compra**: al hold se crea SeatHold (10 min). Si paga en ventana → Ticket.SOLD; si no → HOLD.EXPIRED y silla disponible.
3. **Overbooking**: si ocupación > 95% y faltan <30 min para salir, permitir +1 silla (si overbooking% lo permite) con aprobación DISPATCHER.
4. **Encomienda entregada**: DELIVERED requiere OTP y foto; si OTP falla → FAILED + Incident.
5. **Operación offline**: ventas en taquilla sin red quedan como pendingSync y se reconcilian al volver la conexión.

---

7) Arquitectura recomendada (Spring Boot 3.4.x)

- **Security**: JWT (roles por endpoint).
- **Capas**: Controllers (DTO + Bean Validation) → Services (reglas de tramo, hold, dinámica de precios, overbooking, offline) → Repositories (Spring Data JPA + PostgreSQL). Migraciones con **Flyway**.
- **Mapping**: **MapStruct**.
- **Jobs**: @Scheduled para expirar SeatHold y reportes de puntualidad.

- **Observabilidad**: Micrometer/Actuator; logs estructurados; trazas distribuidas opcionales.
- **Testing**: JUnit5 + Mockito + **Testcontainers** (PostgreSQL); MockMvc para endpoints.

---

8) Paso a paso sugerido

1. Bootstrap Maven + dependencias.
2. Esquema y entidades básicas (Route, Stop, Bus, Trip, SeatHold, Ticket, Parcel).
3. Búsqueda de trips y hold de sillas.
4. Compra de tickets con validación de tramos y pagos mock.
5. Panel de despacho (assign/boarding/depart).
6. Encomiendas con OTP/foto y flujo de entrega.
7. Cierres de caja y métricas para Admin.
8. Pruebas unitarias/web/integración.

---

9) KPIs

- Ocupación por trip (p50/p95), puntualidad (salida/llegada), tasa de no-show.
- Ventas por canal (taquilla/app) y por método de pago.
- Encomiendas entregadas vs fallidas por tramo.

---

10) Errores estándar

Código	Mensaje
400	Validación fallida (tramo inválido, exceso de equipaje)
401	No autenticado
403	Rol no autorizado / política de overbooking
404	Recurso no encontrado

409 Conflicto (silla ocupada/hold activo)

422 Estado inválido para transición

500 Error interno

---

11) Extensiones (para nota extra)

- **Asientos preferenciales** y accesibilidad (bloqueos/reglas).
- **Pricing dinámico** basado en ETA/clima/eventos.
- **Integración con lectura de QR** en abordaje.
- **Simulador de demanda** para pruebas de estrés.

12) Glosario de dominio ampliado

- **Tramo**: par consecutivo de paradas Stop[i] → Stop[i+1]. La disponibilidad de una silla se gestiona **por tramos**.
- **Salida (Trip)**: instancia programada de una Route en una fecha/hora con un bus asignado.
- **Hold**: bloqueo temporal de una silla por 10 min mientras el pasajero paga.
- **Ticket parcial**: boleto válido solo para un subconjunto de tramos entre fromStop y toStop.
- **Overbooking**: venta controlada por encima de la capacidad nominal con políticas.
- **Taquilla offline**: modo de operación sin conectividad con colas de sincronización.

13) Historias de usuario por rol (selección)

PASSENGER

1. Como pasajero, quiero ver salidas disponibles, seleccionar **tramo** y silla en mapa para comprar sin errores de disponibilidad.
2. Como pasajero, quiero **cancelar** con reembolso según antelación.
3. Como pasajero, quiero recibir el **QR del ticket** y notificaciones de cambios de andén.

CLERK (taquilla) 4. Como taquillero, quiero vender en **modo offline** y que el sistema reconcilie luego sin duplicar sillas. 5. Como taquillero, quiero registrar **equipaje** con peso y tag.

DISPATCHER 6. Como despachador, quiero configurar **% overbooking** por ruta/hora y aprobar casos límite. 7. Como despachador, quiero monitorear ocupación en tiempo real por tramos.

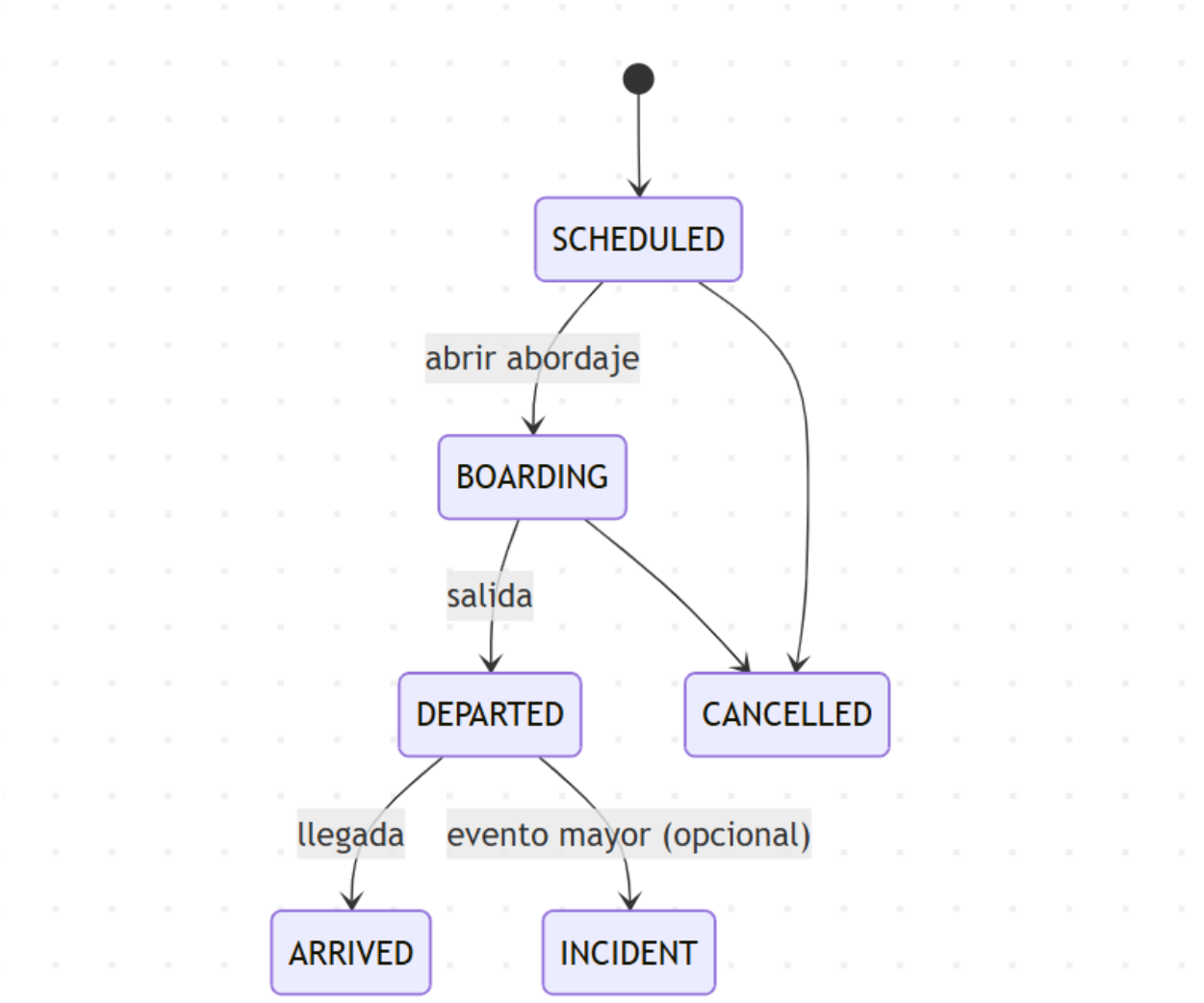
DRIVER 8. Como conductor, quiero validar **QR** en abordaje (online/offline) y ver lista de pasajeros por tramo.

ADMIN 9. Como admin, quiero definir **políticas de reembolso**, descuentos y dinámicas de precio.

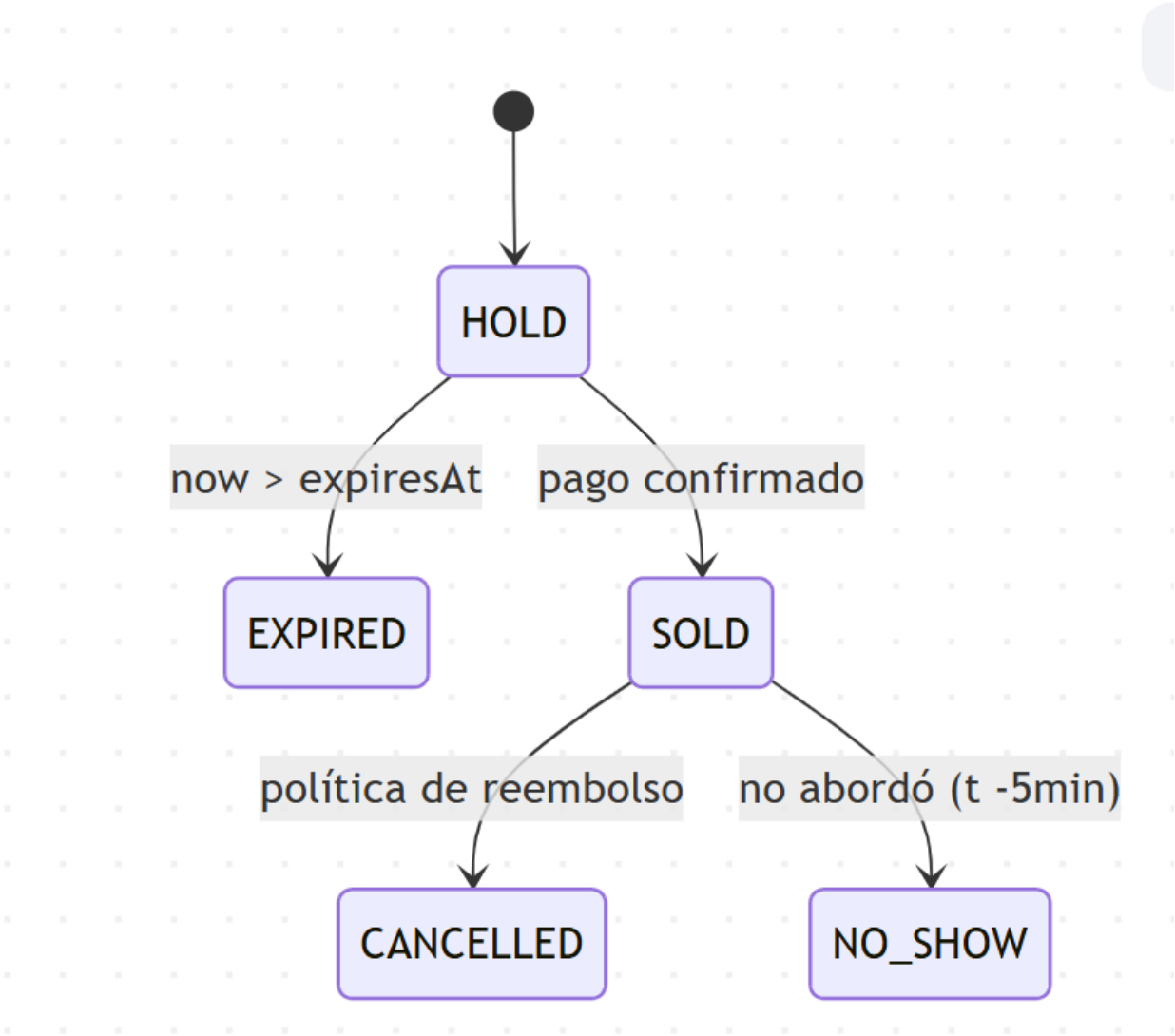
AC: Cada historia debe tener criterios medibles y pruebas MockMvc/Integración.

14) Máquinas de estados

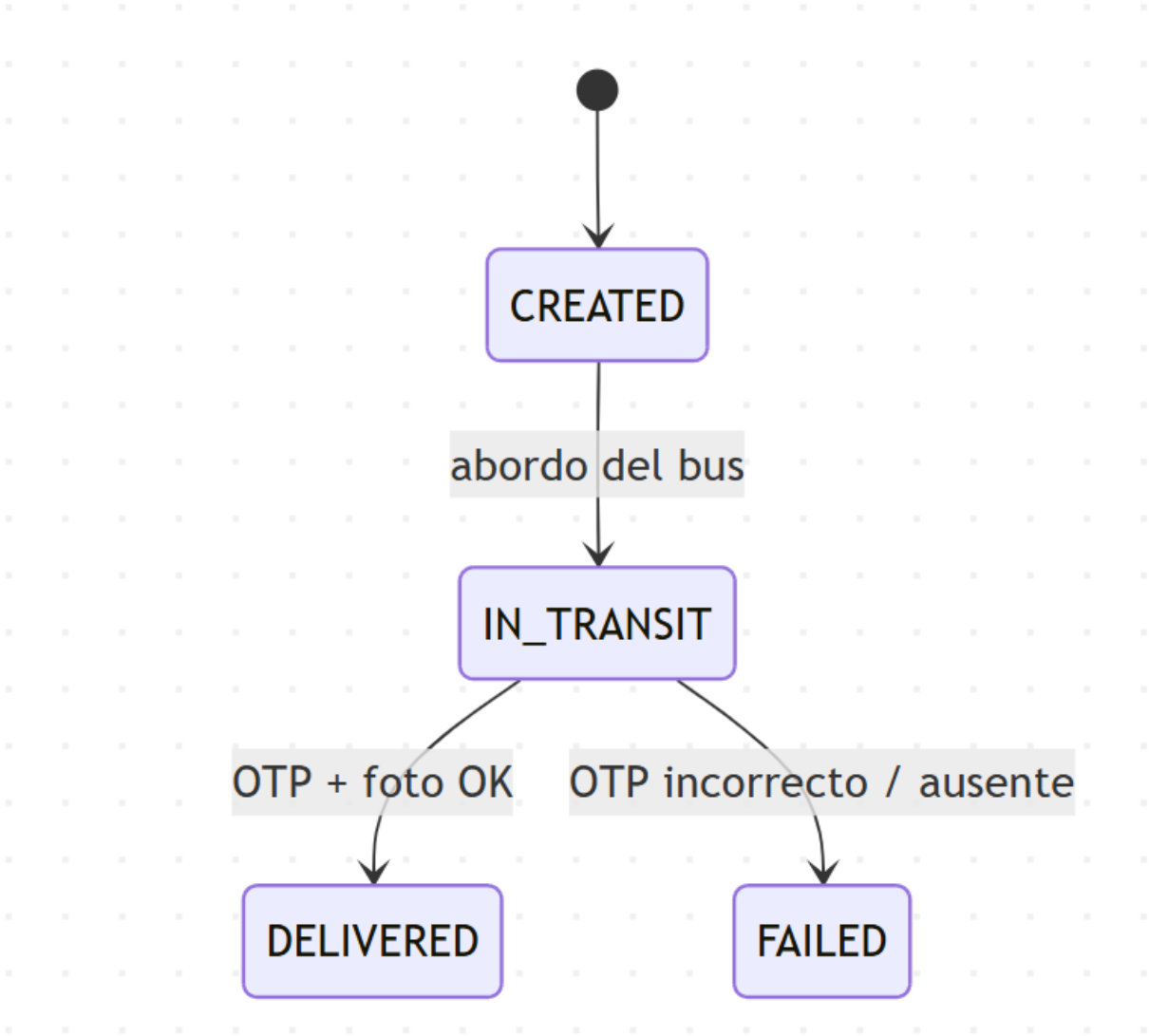
14.1 Trip



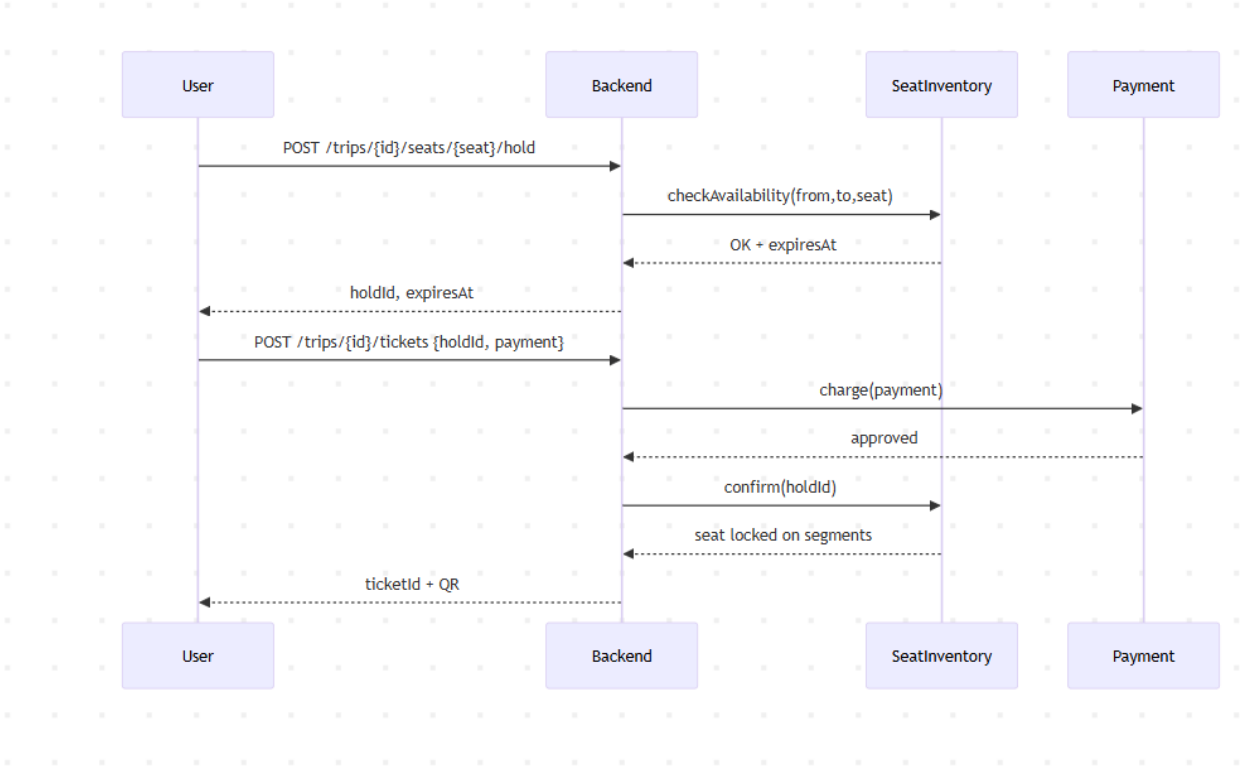
14.2 SeatHold & Ticket



14.3 Parcel (encomienda)



14) Diagrama de secuencia — Hold y Compra



15) Roadmap (iteraciones sugeridas)

1. Iteración 1: Búsqueda de trips, hold, compra, cancelación básica.
2. Iteración 2: Encomiendas con OTP/foto y equipaje.
3. Iteración 3: Dinámica de precios y overbooking con aprobación.
4. Iteración 4: Offline taquilla/driver y reconciliación.
5. Iteración 5: Métricas, reportes y hardening (idempotencia, auditoría, tests de carga).

16) UI

Plataforma Intermunicipal — Demo UI

Busca viajes, elige asiento, paga tu ticket y registra encomiendas. Incluye panel de despacho y cola offline de taquilla.

Taquilla: cola offline (2)

Viajes | Encomiendas | Despacho

Buscar salidas

Selecciona ruta, fecha y tramo (origen → destino)

Ruta	Fecha	Origen	Destino	Tarifa
Bogotá ↔ Tunja	10/25/2025	Bogotá (Terminal)	Tunja (Terminal)	Sin descuento

Bogotá ↔ Tunja 2025-10-25

08:30 → 11:30 · Bus ABC-123 · 40 sillas

Estado: **SCHEDULED** Ver asientos

Bogotá ↔ Tunja 2025-10-25

12:00 → 15:00 · Bus ABC-456 · 40 sillas

Estado: **SCHEDULED** Ver asientos

Bogotá ↔ Tunja 2025-10-25

17:30 → 20:30 · Bus ABC-789 · 40 sillas

Estado: **SCHEDULED** Ver asientos