Booking a Flight - Sequence Diagram

Sequence diagram with lifelines:

- **Customer**
- **PaymentView** «boundary»
- **PaymentService** «control»
- **PaymentDAO** «entity»
- **Payment** «entity»
- **ReservationService** «control»
- **SeatDAO** «entity»
- **Seat** «entity»
- **ReservationDAO** «entity»
- **Reservation** «entity»
- **FlightService** «control»
- **FlightDAO** «entity»
- **Flight** «entity»

Messages:

- Customer → PaymentView: clicks "Pay" button
- PaymentView → PaymentView: getCurrentUser()
- PaymentView → PaymentService: processPayment(amount, method, cardNumber, cardHolder, expiry, cvv)
- PaymentService → PaymentService: validatePaymentDetails(amount, method, cardNumber, cardHolder, expiry, cvv)
- PaymentService → PaymentService: generateTransactionId()
- PaymentService → Payment: new Payment(0, amount, LocalDateTime.now(), method, transactionId, PENDING)
- PaymentService → PaymentService: simulatePaymentAuthorization(cardNumber, amount)

alt [Payment authorized]

- PaymentService → PaymentDAO: save(payment)
- PaymentDAO: INSERT INTO payments
- PaymentDAO → PaymentService: return savedPayment
- PaymentService → Payment: setStatus(COMPLETED)
- PaymentService → PaymentDAO: update(savedPayment)
- PaymentDAO: UPDATE payments SET status = 'COMPLETED'
- PaymentDAO → PaymentService: return true
- PaymentService → PaymentView: return savedPayment
- PaymentView → ReservationService: createReservationForUser(user, flight, selectedSeats, payment)
- ReservationService → ReservationService: createReservationInternal(customer, flight, selectedSeats, payment)
- ReservationService → ReservationService: validate customer, flight, seats not null

loop [for each seat in selectedSeats]

- ReservationService → SeatDAO: findById(seat.getSeatId())
- SeatDAO: SELECT * FROM seats WHERE seat_id = ?
- SeatDAO → ReservationService: return dbSeat
- ReservationService → ReservationService: check if dbSeat == null
- ReservationService → Seat: isAvailable()
- Seat → ReservationService: return boolean
- ReservationService → ReservationService: check if seat is available

- ReservationService → Flight: getAvailableSeats()
- Flight → ReservationService: return availableSeats
- ReservationService → ReservationService: check flight.getAvailableSeats() >= selectedSeats.size()
- ReservationService → ReservationService: calculateTotalPrice(flight, selectedSeats)
- ReservationService → Reservation: new Reservation(0, LocalDateTime.now(), CONFIRMED, totalPrice, customer, flight, payment, selectedSeats)
- ReservationService → ReservationDAO: save(reservation)
- ReservationDAO: INSERT INTO reservations
- ReservationDAO → ReservationService: return savedReservation

loop [for each seat in selectedSeats]

- ReservationService → Seat: setAvailable(false)
- ReservationService → SeatDAO: update(seat)
- SeatDAO: UPDATE seats SET is_available = false WHERE seat_id = ?
- SeatDAO → ReservationService: return true

- ReservationService → Flight: getFlightNumber()
- Flight → ReservationService: return flightNumber
- ReservationService → FlightService: bookSeats(flightNumber, selectedSeats.size())
- FlightService → FlightService: getFlightByNumber(flightNumber)
- FlightService → FlightDAO: findByFlightNumber(flightNumber)
- FlightDAO: SELECT * FROM flights WHERE flight_number = ?
- FlightDAO → FlightService: return flight
- FlightService → Flight: getAvailableSeats()
- Flight → FlightService: return availableSeats
- FlightService → Flight: setAvailableSeats(availableSeats - seatsBooked)
- FlightService → FlightDAO: update(flight)
- FlightDAO: UPDATE flights SET available_seats = ? WHERE flight_id = ?
- FlightDAO → FlightService: return true
- FlightService → ReservationService: return true
- ReservationService → PaymentView: return savedReservation
- PaymentView → PaymentView: showView("CONFIRMATION", ConfirmationView(reservation))
- PaymentView → Customer: display booking confirmation

[Payment authorization failed]

- PaymentService → Payment: setStatus(FAILED)
- PaymentService → PaymentDAO: save(payment)
- PaymentDAO: INSERT INTO payments
- PaymentDAO → PaymentService: return savedPayment
- PaymentService → PaymentView: throw IllegalStateException("Payment authorization failed...")
- PaymentView → Customer: display payment failed error