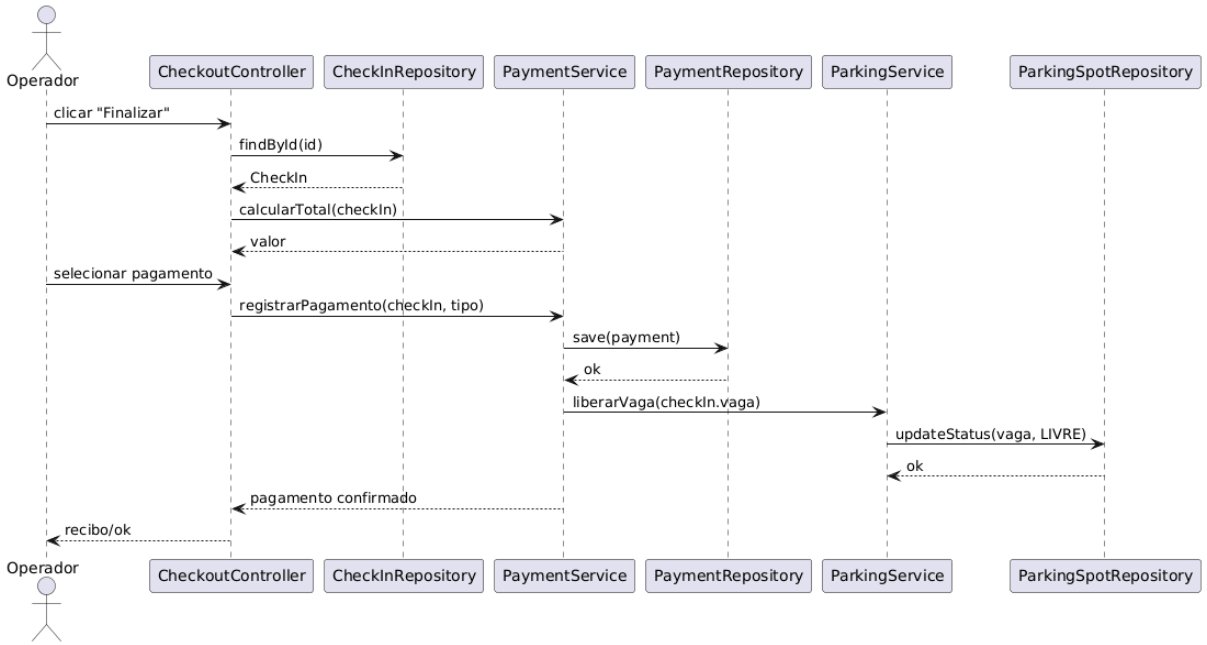
Modelagem de Interações



A modelagem de interações descreve a colaboração dinâmica entre objetos do sistema EasyStop ao longo dos principais fluxos de operação.
Como o sistema segue uma arquitetura **MVC + Repositórios + Serviços**, os diagramas de sequência focam na troca de mensagens entre:

- **Interface (FXML + Controllers JavaFX)**
- **Modelos de domínio (Vehicle, ParkingSpot, CheckIn, Payment)**
- **Serviços encapsulando regras (ParkingService, PaymentService)**
- **Repositórios ORMLite**

A seguir são apresentados os três fluxos principais e seus diagramas.

Fluxo 1 — Check-in de Veículo

Descrição Resumida

Este fluxo ocorre quando o operador registra a entrada de um veículo, vinculando-o a uma vaga disponível e armazenando o check-in no banco via ORMLite.

Participantes

Papel	Classe
UI Controller	`CheckInController`
Serviço	`ParkingService`
Repositórios	`VehicleRepository`, `ParkingSpotRepository`, `CheckInRepository`
Modelos	`Vehicle`, `ParkingSpot`, `CheckIn`

Diagrama UML (PlantUML)

```
@startuml
actor Operador
Operador -> CheckInController: clicar "Registrar Check-in"
CheckInController -> VehicleRepository: findOrCreate(placa)
VehicleRepository --> CheckInController: Vehicle

CheckInController -> ParkingSpotRepository: getAvailable()
ParkingSpotRepository --> CheckInController: ParkingSpot

CheckInController -> ParkingService: createCheckIn(vehicle, vaga)
ParkingService -> CheckInRepository: save(checkIn)
CheckInRepository --> ParkingService: ok

ParkingService -> ParkingSpotRepository: updateStatus(vaga, OCUPADA)
ParkingSpotRepository --> ParkingService: ok

ParkingService --> CheckInController: checkIn registrado
CheckInController --> Operador: confirmação
@enduml
```

Fluxo 2 — Checkout + Pagamento

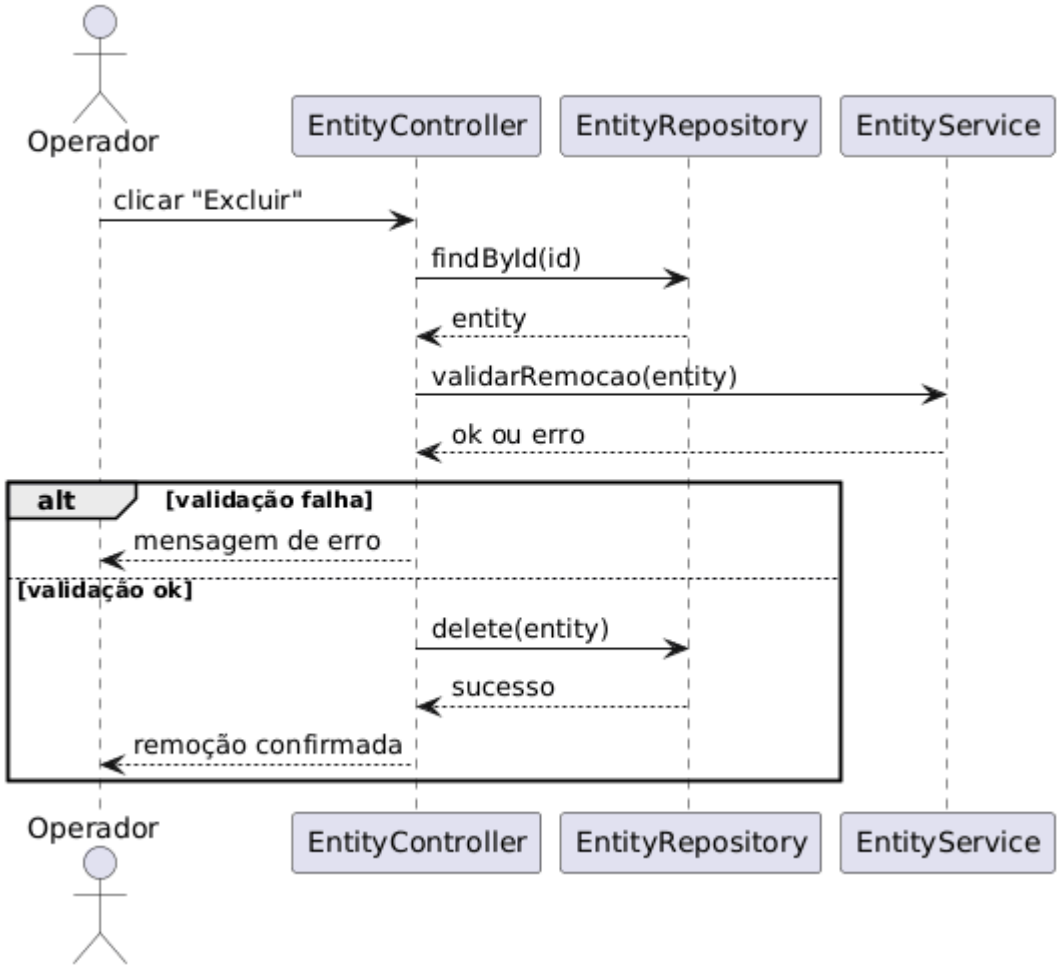
Descrição Resumida

O sistema encerra um check-in, calcula o valor, solicita tipo de pagamento e finaliza o processo gravando pagamento e liberando a vaga.

Participantes

Papel	Classe
Controller	`CheckoutController`
Serviço	`PaymentService`, `ParkingService`
Repositórios	`CheckInRepository`, `PaymentRepository`, `ParkingSpotRepository`
Modelos	`CheckIn`, `Payment`, `ParkingSpot`

Fluxo 3 — Deleção de Registro (Veículo, Vaga ou Check-in)



Racional

A deleção é sensível pois envolve integridade de dados.
No EasyStop, a regra geral é:

- **Não deletar check-in em andamento**
- **Não deletar vaga ocupada**
- **Não quebrar ligação cascata sem antes validar**

A interação abaixo descreve o fluxo genérico aplicado ao padrão da sua aplicação.

Participantes

Papel	Classe
Controller	`<Entity>Controller` (ex: `VehicleController`)
Serviço	`<Entity>Service` quando existir
Repositório	`<Entity>Repository`
Modelo	`Vehicle` / `ParkingSpot` / `CheckIn`