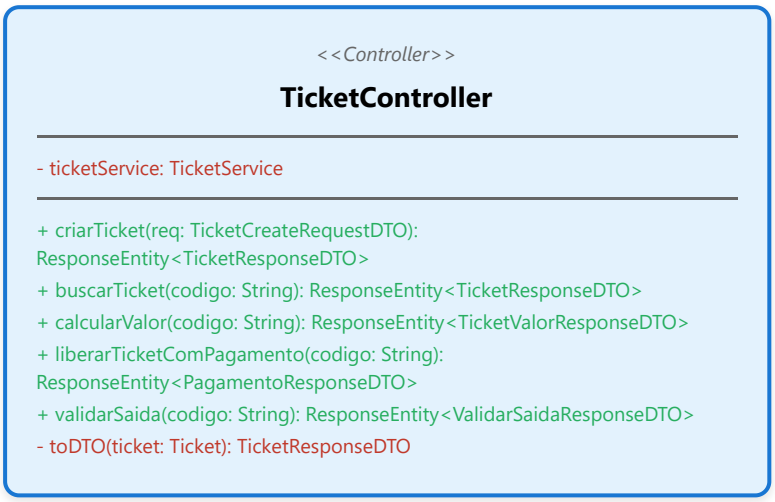
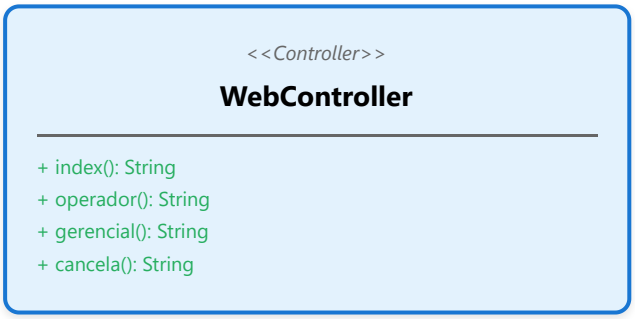
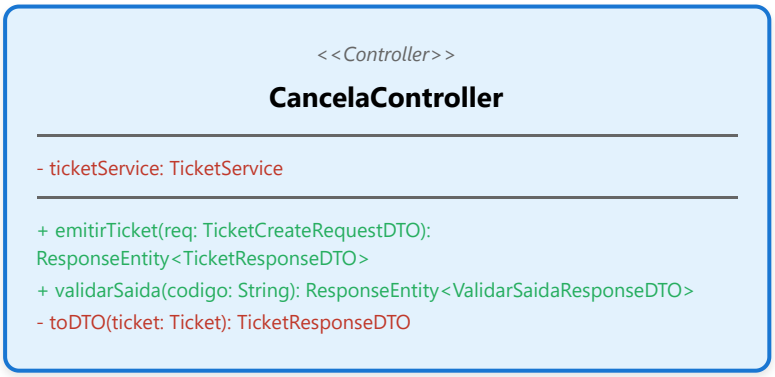
📐 Diagrama de Classes - Sistema de Estacionamento

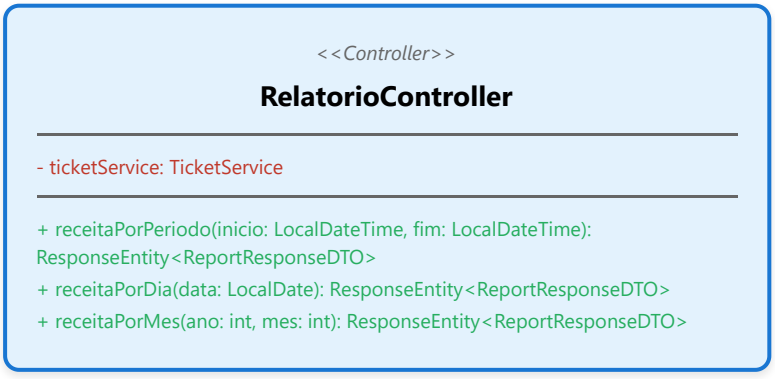
Padrões: MVC + Domain Model + Repository Pattern

🧮 Camada de Apresentação (Controllers)









⚙️ Camada de Negócio (Services)

<<Service>>

TicketService

- ticketRepository: TicketRepository
- pagamentoRepository: PagamentoRepository

+ criarTicket(placa: String): Ticket
+ calcularValor(codigo: String): BigDecimal
+ validarSaida(codigo: String): boolean
+ pagarTicket(codigo: String): boolean
+ totalRecebido(inicio: LocalDateTime, fim: LocalDateTime): BigDecimal
+ contarTicketsPagos(inicio: LocalDateTime, fim: LocalDateTime): Integer
+ buscarPorCodigo(codigo: String): Optional<Ticket>
+ pagarETrazerPagamento(codigo: String): Pagamento

🏗️ Camada de Domínio (Entities)

<<Entity>>

Ticket

- codigo: String
- placa: String
- entrada: LocalDateTime
- saida: LocalDateTime
- pago: boolean
- valor: BigDecimal

+ Ticket()
+ Ticket(codigo: String, placa: String, entrada: LocalDateTime)
+ getCodigo(): String
+ getPlaca(): String
+ getEntrada(): LocalDateTime
+ getSaida(): LocalDateTime
+ isPago(): boolean
+ getValor(): BigDecimal
+ isVencido(): boolean
+ isPeriodoCortesia(): boolean
+ marcarComoPago(valorPago: BigDecimal): void
+ registrarSaida(): void

<<Entity>>

Pagamento

- id: Long
- ticketCodigo: String
- valor: BigDecimal
- dataPagamento: LocalDateTime

+ Pagamento()
+ Pagamento(ticketCodigo: String, valor: BigDecimal)
+ getId(): Long

+ getTicketCodigo(): String
+ getValor(): BigDecimal
+ getDataPagamento(): LocalDateTime

💾 Camada de Persistência (Repositories)

<<Repository>>

TicketRepository

extends JpaRepository<Ticket, String>

+ findByPagoTrueAndSaidaBetween(inicio: LocalDateTime, fim: LocalDateTime): List<Ticket>
+ findByPagoTrueAndSaidaBetweenOrderBySaidaAsc(inicio: LocalDateTime, fim: LocalDateTime): List<Ticket>
+ findByPlaca(placa: String): List<Ticket>

<<Repository>>

PagamentoRepository

extends JpaRepository<Pagamento, Long>

+ findByDataPagamentoBetween(inicio: LocalDateTime, fim: LocalDateTime): List<Pagamento>

📦 Data Transfer Objects (DTOs)

<<DTO>>

TicketResponseDTO

- codigo: String
- placa: String
- entrada: LocalDateTime
- saida: LocalDateTime
- pago: boolean
- valor: BigDecimal

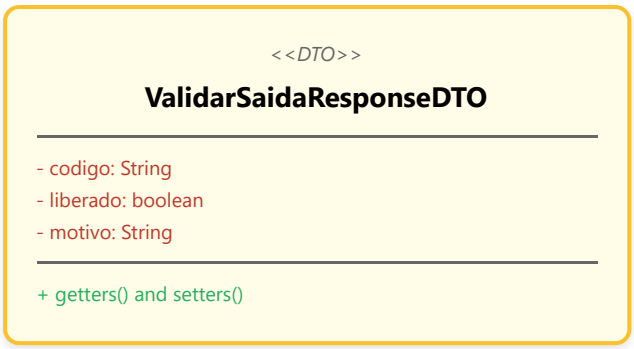
+ getters() and setters()

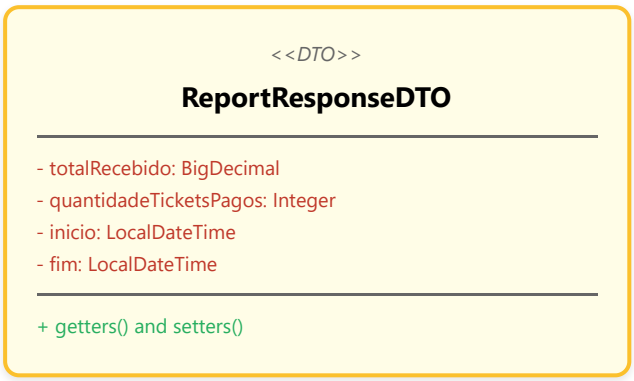
<<DTO>>

TicketCreateRequestDTO

- placa: String

+ getPlaca(): String
+ setPlaca(placa: String): void





🔗 Relacionamentos Entre Classes

TicketController → **TicketService** (usa)

CancelaController → **TicketService** (usa)

RelatorioController → **TicketService** (usa)

TicketService → **TicketRepository** (usa)

TicketService → **PagamentoRepository** (usa)

TicketService → **Ticket** (gerencia)

TicketService → **Pagamento** (gerencia)

TicketRepository → **Ticket** (persiste)

PagamentoRepository → **Pagamento** (persiste)

Ticket (1) → **Pagamento** (0..*) (relacionamento lógico)

✅ Padrões de Projeto Implementados

MVC (Model-View-Controller)
Onde: Camada de Apresentação
Como: Controllers processam requisições HTTP, Services contêm lógica de negócio, DTOs transportam dados

Domain Model
Onde: Entities (Ticket, Pagamento)
Como: Objetos ricos com comportamento e validações de negócio encapsuladas

Repository Pattern
Onde: Camada de Persistência

Como: Abstração do acesso a dados através de interfaces JPA

Dependency Injection
Onde: Em todas as camadas
Como: Spring Framework gerencia dependências via constructor injection

Data Transfer Object (DTO)
Onde: Comunicação entre camadas
Como: Objetos simples para transporte de dados sem lógica de negócio