



Diagrama de Classes - Sistema de Estacionamento

Padrões: MVC + Domain Model + Repository Pattern



Camada de Apresentação (Controllers)

<<Controller>>

TicketController

- ticketService: TicketService

+ criarTicket(req: TicketCreateRequestDTO):
ResponseEntity<TicketResponseDTO>
+ buscarTicket(codigo: String): ResponseEntity<TicketResponseDTO>
+ calcularValor(codigo: String): ResponseEntity<TicketValorResponseDTO>
+ liberarTicketComPagamento(codigo: String):
ResponseEntity<PagamentoResponseDTO>
+ validarSaida(codigo: String): ResponseEntity<ValidarSaidaResponseDTO>
- toDTO(ticket: Ticket): TicketResponseDTO

<<Controller>>

WebController

+ index(): String
+ operador(): String
+ gerencial(): String
+ cancela(): String

<<Controller>>

CancelaController

- ticketService: TicketService

+ emitirTicket(req: TicketCreateRequestDTO):
ResponseEntity<TicketResponseDTO>
+ validarSaida(codigo: String): ResponseEntity<ValidarSaidaResponseDTO>
- toDTO(ticket: Ticket): TicketResponseDTO

<<Controller>>

RelatorioController

- ticketService: TicketService

+ receitaPorPeriodo(inicio: LocalDateTime, fim: LocalDateTime):
ResponseEntity<ReportResponseDTO>
+ receitaPorDia(data: LocalDate): ResponseEntity<ReportResponseDTO>
+ receitaPorMes(ano: int, mes: int): ResponseEntity<ReportResponseDTO>

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
```

<<DTO>>

ValidarSaidaResponseDTO

- codigo: String
- liberado: boolean
- motivo: String

+ getters() and setters()

<<DTO>>

ReportResponseDTO

- totalRecebido: BigDecimal
- quantidadeTicketsPagos: Integer
- inicio: LocalDateTime
- fim: LocalDateTime

+ getters() and setters()

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



Controller Layer



Service Layer



Domain Layer



Repository Layer



DTO Layer