

Modelagem de Estados — EasyStop

Este documento apresenta a modelagem de estados das principais entidades do sistema **EasyStop**, com base na implementação atual. A modelagem foca na identificação de atributos que representam estado, nas transições possíveis e nas condições que governam essas mudanças. O objetivo é evidenciar o comportamento dinâmico do sistema ao longo do ciclo de vida de seus objetos.

1. Introdução

A modelagem de estados descreve como entidades mudam ao longo do tempo, reagindo a eventos gerados pelo usuário, por regras de negócio ou por processos internos do sistema. No EasyStop, diversas funcionalidades dependem do estado atual de objetos como **Check-in**, **Vaga**, **Pagamento** e os controladores CRUD.

O sistema utiliza:

- **Atributos booleanos** para representar estados simples.
- **Enums** para representar estados compostos e finitos.
- **Controladores** que implementam transições condicionais e validam o ciclo de vida das entidades.

A seguir, apresentamos as máquinas de estados de cada entidade.

2. Máquina de Estado — Check-in

Atributo de Estado

- `boolean finalizado`

Estados Possíveis

| Estado | Descrição |
|---|--|
| Ativo (<code>finalizado = false</code>) | Check-in criado, ainda não finalizado. |

Finalizado (`finalizado = true`) Check-in finalizado após checkout.

Eventos que Alteram Estado

| Evento | Origem | Estado Final | Condições |
|----------------------|---|--------------|---|
| Criar Check-in | <code>CheckinController.doCreate()</code> | Ativo | — |
| Finalizar Check-in | <code>CheckoutController.finalizarCheckin()</code> | Finalizado | Pagamento aprovado |
| Reverter Finalização | <code>CheckoutController.desfazerFinalizacao()</code> | Ativo | O checkout é excluído antes da finalização permanente |

Restrições de Estado

- Um check-in **finalizado não pode ser editado**.
- Um check-in **finalizado não pode ser deletado**.
- Mensagens de alerta são exibidas se tentado.

3. Máquina de Estado — Vaga

Atributo de Estado

- `boolean ocupada`

Estados Possíveis

| Estado | Descrição |
|---|--------------------------------|
| Livre (<code>ocupada = false</code>) | Vaga disponível para check-in. |

Ocupada (`ocupada = true`) Vaga em uso por um veículo.

Eventos que Alteram Estado

| Evento | Origem | Estado Final |
|-----------------------------------|--|--------------|
| Check-in criado | <code>CheckinController.afterCreate()</code> | Ocupada |
| Finalizar Checkout | <code>CheckoutController.finalizarCheckin()</code> | Livre |
| Deletar Checkout (não finalizado) | <code>desfazerFinalizacao()</code> | Ocupada |
| Deletar Check-in ativo | <code>onDeletear()</code> | Livre |

Transições Relevantes

- A vaga retorna para **livre** sempre que um check-in é removido ou finalizado.
- A vaga vai para **ocupada** durante criação do check-in.

4. Máquina de Estado — Pagamento

Atributo de Estado

- `StatusPagamento status`

Enum de Estado

- PENDENTE
- APROVADO
- RECUSADO

Estados Possíveis

| Estado | Significado |
|----------|--------------------------------------|
| Pendente | Pagamento criado mas não processado. |
| Aprovado | Pagamento validado e autorizado. |
| Recusado | Pagamento inválido ou insuficiente. |

Eventos que Alteram Estado

| Evento | Classe | Estado Final |
|----------------------------------|---|---------------------|
| Processar pagamento via cartão | <code>PagamentoCartao.processarPagamento()</code> | APROVADO / RECUSADO |
| Processar pagamento via dinheiro | <code>PagamentoDinheiro.processarPagamento()</code> | APROVADO / RECUSADO |
| Processar pagamento via PIX | <code>PagamentoPix.processarPagamento()</code> | APROVADO / RECUSADO |
| Atualizar status manualmente | <code>PagamentoRepositorio.updateStatus()</code> | Qualquer estado |

Condições Importantes

- Checkout **só pode ser criado** se o pagamento estiver **APROVADO**.
- Um pagamento **não pode ser reprocessado** pelas regras atuais, mas pode ser reeditado via controller.

5. Máquina de Estado — Controlador CRUD

Atributo de Estado

- `Action pendingAction`

Enum Action

- NONE
- NOVO
- ATUALIZAR
- DELETAR

Estados da UI

| Estado | Descrição |
|-----------|---|
| NONE | Interface em repouso; nenhuma ação ativa. |
| NOVO | Usuário criando entidade. |
| ATUALIZAR | Usuário editando entidade selecionada. |
| DELETAR | Usuário deletando entidade selecionada. |

Transições

| Evento | Estado Final |
|--------------------|---|
| Clicar “Adicionar” | NOVO |
| Clicar “Atualizar” | ATUALIZAR |
| Clicar “Deletar” | DELETAR |
| Confirmar ação | NONE |
| Cancelar ação | NONE |
| Selecionar item | Permite ações somente se estado = NONE |

Impacto na Interface

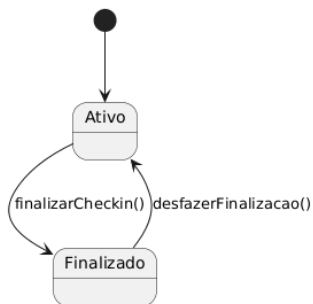
Durante **NOVO**, **ATUALIZAR**, **DELETAR**:

- Campos são habilitados/desabilitados.
- Botões de CRUD são bloqueados.
- Tabela é congelada para evitar troca de seleção.

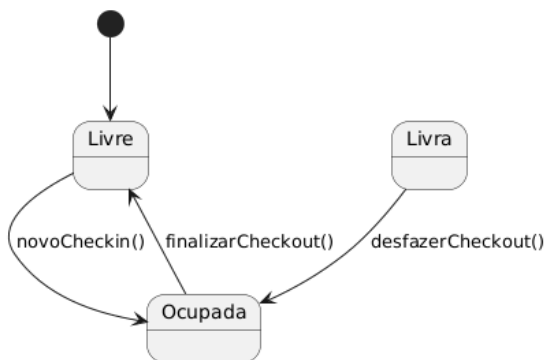
6. Resumo da Modelagem de Estados

| Entidade | Estados | Transições | Controlador |
|-----------|-----------------------------------|----------------------|--|
| Check-in | Ativo / Finalizado | Finalizar / Reverter | CheckinController & CheckoutController |
| Vaga | Livre / Ocupada | Ocupar / Liberar | CheckinController & CheckoutController |
| Pagamento | Pendente / Aprovado / Recusado | Processar | PagamentoService & Controllers |
| UI (CRUD) | NONE / NOVO / ATUALIZAR / DELETAR | Comandos CRUD | AbstractCrudController |

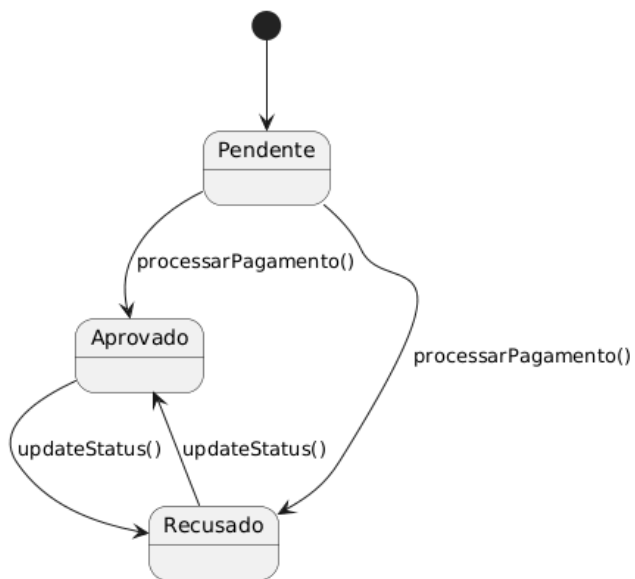
Estado — Check-in



Estado — Vaga



Estado — Pagamento



Estado — Ciclo CRUD da UI

