

# POO 2025/2 — Atividade Única: Associações

Material único que consolida **mini-aplicações web visuais** e **10 situações por atividade** para um workshop/seminário sobre **associações entre classes**. Todos os exemplos são diferentes do material-base, com validação argumentativa fundamentada nos conceitos de multiplicidade, opcionalidade, navegabilidade mínima, classe de associação, composição × agregação, qualificador, cheiros e refatorações.

## Visão Geral do Workshop

### Mini-aplicações Web

Interface visual interativa com UI/fluxo específico para cada atividade

- Client-side (HTML/CSS/JS)
- Exportação PNG e JSON/CSV
- Sem necessidade de backend

### 10 Situações Práticas

Cenários reais organizados por níveis de dificuldade

- Elementar (E): 3 situações
- Intermediário (I): 4 situações
- Avançado (A): 3 situações

### Validação

#### Argumentativa

Sem gabarito fixo - fundamentação conceitual

- Debate estruturado
- Defesa de escolhas
- Reflexão crítica

## Atividade 1: Reconhecendo Associações

### Objetivo

Destacar numa narrativa curta **candidatos a classes** (substantivos) e **pistas de vínculo** (verbos/preposições), rabiscando associações iniciais.

### Organização dos Grupos (3-5 pessoas)

- Leitor(a) da história
- Mapeador(a) de classes
- Detector(a) de vínculos
- Cronometrista/porta-voz

### Tempo de Execução

3-5 minutos por história no seminário



# Mini-app: Detector de Associações

01	02	03
<b>Entrada de Dados</b>	<b>Análise Textual</b>	<b>Montagem Visual</b>
Textarea "Cole a história" para inserir a narrativa a ser analisada	Botões "Destacar substantivos" e "Destacar verbos/preposições" com cores diferentes	Painel de chips arrastáveis para montar pares "Classe A — (verbo/preposição)→ Classe B"
04	05	
<b>Visualização</b>	<b>Exportação</b>	
Canvas que desenha nós e arestas representando o grafo de classes	PNG do canvas + JSON com classes e associações identificadas	

## 10 Histórias para Análise

1	<b>[E] Pet Shop</b> Um pet shop agenda banhos e tosas para tutores e entrega senhas de atendimento.
2	<b>[E] Paraciclo Público</b> Um paraciclo público registra entradas e saídas de bicicletas usando QR codes.
3	<b>[E] Coworking</b> Um coworking vende passes diários e libera vouchers de Wi-Fi para visitantes.
4	<b>[I] Plataforma de Entrega</b> Uma plataforma de entrega conecta restaurantes e entregadores por meio de pedidos.
5	<b>[I] Clínica de Vacinas</b> Uma clínica de vacinas agenda aplicações e emite comprovantes para clientes.

# Mais Histórias Complexas

1

## [I] Parque Temático

Um parque temático usa pulseiras RFID para visitantes acessarem atrações.

2

## [I] Oficina de Drones

Uma oficina de drones registra serviços e peças usadas em cada reparo.

3

## [A] Operadora de Energia

Uma operadora de energia monitora medidores inteligentes e emite faturas com tarifas por faixa.

4

## [A] Porto

Um porto organiza contêineres em navios segundo janelas de atracação e documentos alfandegários.

5

## [A] Rede Ambiental

Uma rede ambiental coleta leituras de sensores e notifica autoridades com alertas.

## Atividade 2: Associação ou Atributo?

### Objetivo

Decidir quando algo é **atributo** ou deve virar **classe associada** considerando identidade, regras, histórico e ciclo de vida.

### Organização dos Grupos (4 pessoas)

- Dupla "atributo"
- Dupla "classe"
- Convergência e defesa final

### Critérios de Decisão

- Varia no tempo?
- Tem regras próprias?
- Possui histórico?
- Tem identidade própria?
- Ciclo de vida independente?

# Mini-app: Card-Sorting com Radar



## Card-Sorting

Duas colunas: "Atributo" vs "Classe" para arrastar os cartões



## Perguntas-Gatilho

Cada card abre modal com 5 critérios de decisão (toggle)



## Radar Visual

Ao soltar, mostra radar (SVG) com critérios marcados



## Exportação

Decisão + justificativas em formato JSON

## 10 Situações: Atributo ou Classe?

### Cor de Embalagem

De um **produto** - simples propriedade ou entidade com regras?

### Estilo de Preparo

De um **prato** - apenas descrição ou classe com instruções?

### Endereço de Entrega

Com **histórico** por **cliente** - atributo ou entidade temporal?

### Canal de Aquisição

De um **cliente** (orgânico, indicação, campanha X)

### Nível de Risco

Em um **investimento** - propriedade ou classe com cálculos?

### Idioma preferido

De um **usuário**

### Plano de manutenção

De um **equipamento** (calendário/etapas).

### Tipo de entrega

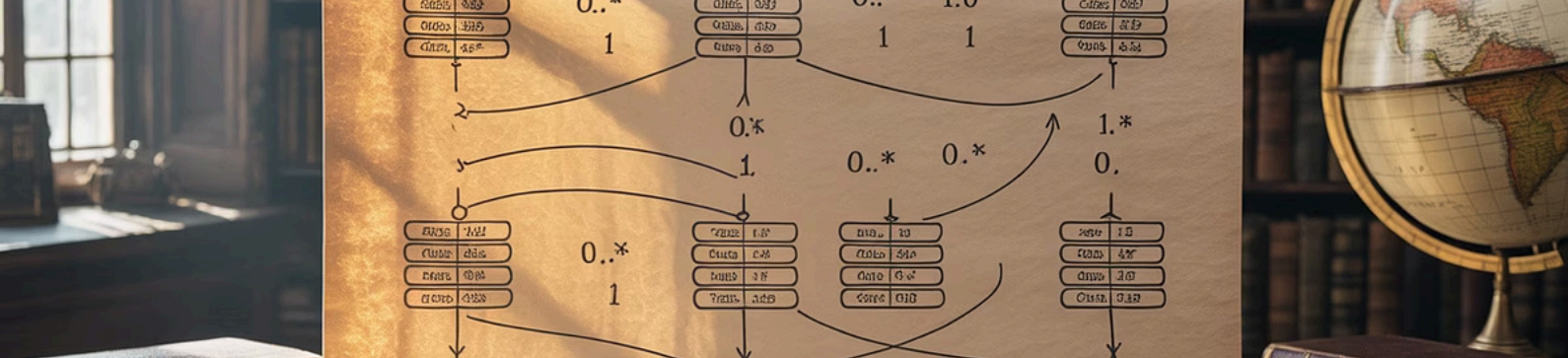
Em uma **compra** (expresso, normal, agendado)

### Política de devolução

De um **vendedor** (regras e prazos).

### Certificado de calibração

De um **instrumento** (número, validade, órgão).



## Atividade 3: Multiplicidade e Opcionalidade

### Objetivo

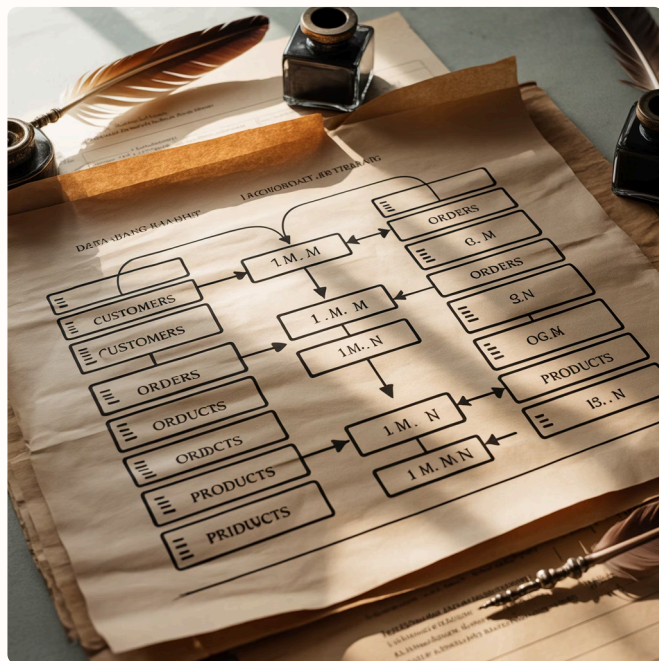
Definir **cardinalidades** (1..1, 0..1, 0..\*, 1..\*) e **opcionalidade** coerentes com o domínio.

### Organização

**Duplas:** Dupla A propõe; Dupla B revisa/desafia

### Cardinalidades Disponíveis

- 0..1 - Opcional, no máximo um
- 1..1 - Obrigatório, exatamente um
- 0..\* - Opcional, qualquer quantidade
- 1..\* - Obrigatório, pelo menos um



## Mini-app: Simulador de Multiplicidade

01

### Configuração

Tabela de pares "A — B" com seletores de multiplicidade para cada lado

02

### Validação

Validador de invariantes com regras textuais curtas do domínio

03

### Simulação

"Instanciar 10 casos aleatórios" e listar violações encontradas

04

### Ajuste

Refinar multiplicidades com base nas violações detectadas

**Uso no seminário:** Configurar 3 pares → simular → mostrar violações → ajustar



# Situações de Multiplicidade

1

## Podcast ↔ Episódios

Um podcast pode ter quantos episódios? Um episódio pertence a quantos podcasts?

2

## Usuário ↔ Métodos de Pagamento

Quantos métodos um usuário pode ter? Um método pode ser de quantos usuários?

3

## Reserva ↔ Passageiros

Uma reserva de viagem pode ter quantos passageiros nomeados?

4

## Quadro Kanban ↔ Cartões

Quantos cartões um quadro pode ter? Um cartão pode estar em quantos quadros?

5

## Vaga ↔ Veículo

Uma vaga de estacionamento pode ter quantos veículos no momento?

6

## Curso EAD ↔ tutores de apoio

Um curso EAD pode ter quantos tutores de apoio? Um tutor pode atender a quantos cursos simultaneamente?

7

## Loja ↔ vitrines sazonais

Uma loja pode ter quantas vitrines sazonais ativas por período? Uma vitrine pode pertencer a quantas lojas ao mesmo tempo?

8

## Voo ↔ bagagens despachadas

Um voo pode ter quantas bagagens despachadas? Uma bagagem pode estar vinculada a quantos voos?

9

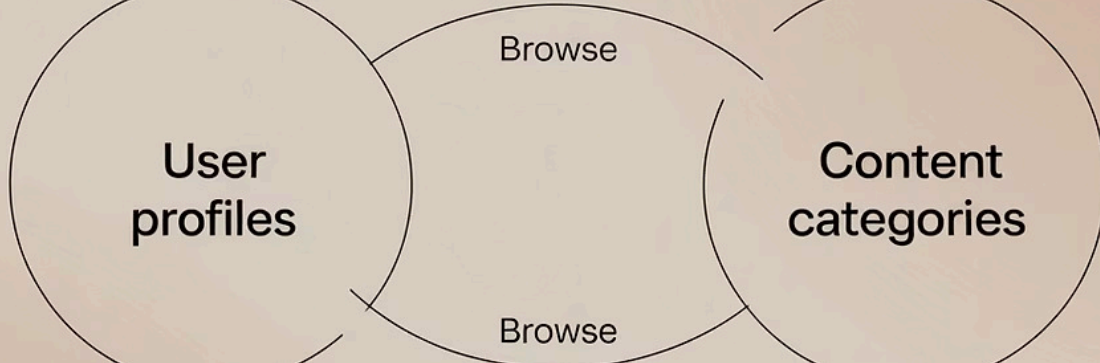
## Projeto ↔ marcos (milestones)

Um projeto pode ter quantos marcos? Um marco pode pertencer a quantos projetos?

10

## Máquina ↔ sensores instalados.

Uma máquina pode ter quantos sensores instalados? Um sensor pode estar instalado em quantas máquinas ao mesmo tempo?



## Atividade 4: Navegabilidade Mínima

### Objetivo

Escolher **somente** os sentidos essenciais de navegação, evitando bidirecional sem necessidade.

### Organização (3 pessoas)

- Defensor  $A \rightarrow B$
- Defensor  $B \rightarrow A$
- Moderador(a)

### Critérios de Decisão

- Uso real da navegação
- Redução de acoplamento
- Leitura dominante
- Complexidade mínima

## Mini-app: Editor de Navegabilidade



### Editor de Grafo

Canvas HTML5 com setas alternáveis (' $\rightarrow$ ', ' $\leftarrow$ ', ' $\times$ ')



### Medidor de Complexidade

Contador de setas e densidade do grafo



### Dicas Inteligentes

"Você realmente precisa da seta oposta?"



### Objetivo

Atingir o mínimo justificável de navegações

## Situações de Navegabilidade Mínima

A seguir, são apresentadas 10 situações para a atividade de navegação, desafiando os grupos a escolherem os sentidos essenciais de navegação para cada par de entidades, justificando suas escolhas com base nos critérios de decisão estabelecidos.

1

**Catálogo ↔ Produtos**

Um **catálogo** exibe muitos **produtos**. É essencial navegar de um produto de volta para o catálogo que o contém, ou a navegação do catálogo para o produto é suficiente?

2

**Remessa ↔ Eventos de Rastreio**

Uma **remessa** precisa listar seus **eventos de rastreio**. É necessário que cada evento de rastreio aponte de volta para a remessa à qual pertence?

3

**Evento ↔ Local**

Um **evento** acontece em um **local** específico. O local precisa ter conhecimento de todos os eventos que já foram ou serão realizados nele?

4

**Receita Médica ↔ Medicações**

Uma **receita médica** prescreve várias **medicações**. Cada medicação precisa ter um registro das receitas nas quais foi prescrita?

5

**Ingresso ↔ Comprador**

Um **ingresso** é emitido para um **comprador**. O comprador precisa de uma lista de todos os ingressos que já adquiriu, ou o ingresso referenciar o comprador é o bastante?

6

**Aplicativo ↔ Permissões Concedidas**

Um **aplicativo** requer **permissões**. As permissões concedidas precisam saber a qual aplicativo foram dadas, além do aplicativo saber quais permissões possui?

7

**Relatório ↔ Autores**

Um **relatório** tem um ou mais **autores**. Um autor precisa ter acesso a todos os relatórios que ele co-autoria, ou a navegação do relatório para o autor é suficiente?

8

**Chamado de Suporte ↔ Anexos**

Um **chamado de suporte** pode ter múltiplos **anexos**. Cada anexo precisa obrigatoriamente referenciar o chamado ao qual está vinculado?

9

**Jornada de Compra ↔ Passos**

Uma **jornada de compra** é composta por uma sequência de **passos**. Um passo individual precisa saber a qual jornada de compra ele pertence?

10

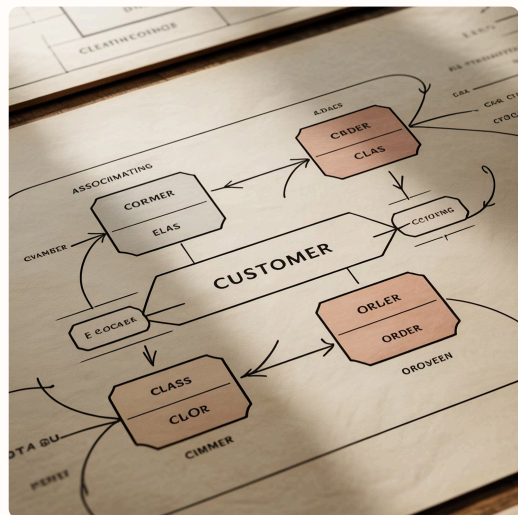
**Contrato ↔ Aditivos**

Um **contrato** pode ter vários **aditivos** ao longo do tempo. É necessário que cada aditivo aponte de volta para o contrato principal?



Ao analisar estas situações, considere o **uso real da navegação**, a **redução de acoplamento** entre as classes, qual é a **leitura dominante** no sistema e como alcançar a **complexidade mínima** justificável.

## Atividade 5: Classe de Associação



### Objetivo

Identificar quando o **vínculo** exige virar **entidade** devido a atributos, regras ou ciclo de vida próprios.

### Organização (4 pessoas)

- 2 pessoas listam atributos do vínculo
- 2 pessoas definem ciclo de vida/responsabilidades

### Sinais de Classe de Associação

- Vínculo tem atributos próprios
- Regras específicas do relacionamento
- Ciclo de vida independente
- Identidade própria do relacionamento

## Mini-App: Editor de Classes de Associação

Este mini-aplicativo foi desenvolvido para auxiliar na compreensão e aplicação do conceito de **Classes de Associação** em diagramas UML, promovendo uma análise visual e interativa dos vínculos entre entidades.

### Fluxo de Interação Intuitivo

O editor permite a definição de um vínculo entre duas entidades (A e B). Um painel dedicado para "Atributos do Vínculo" possibilita a inserção de características ou regras específicas que pertencem à relação, e não às entidades individuais.

### Promoção Dinâmica

Ao detectar a presença de um ou mais atributos ou regras associados diretamente ao vínculo, o mini-aplicativo automaticamente propõe a **promoção** desse vínculo para uma **Classe de Associação**. Isso é visualizado pela inserção de um nó intermediário entre as entidades A e B.

### Comparativo "Antes/Depois"

Para facilitar o aprendizado, o editor apresenta uma visualização clara do diagrama "antes" e "depois" da promoção do vínculo. Além do diff visual, uma comparação textual detalhada explica as mudanças estruturais e semânticas, reforçando o entendimento da necessidade da classe de associação.

### Exportação e Reutilização

O resultado do diagrama, incluindo a nova classe de associação e seus atributos, pode ser exportado em formato JSON, permitindo a integração com outras ferramentas ou a documentação para referência futura.

O aplicativo oferece uma interface interativa que simula a transição de um simples relacionamento para uma classe de associação, tornando o conceito abstrato mais tangível. É uma ferramenta ideal para demonstrações em seminários, onde a justificativa da promoção do vínculo pode ser visualmente apoiada e debatida.

## Situações de Classe de Associação

### Influencer ↔ Campanha

Fee, período, KPI - dados específicos da participação

### Paciente ↔ Exame

Data, preparo, resultado, laudo - informações do procedimento

### Passageiro ↔ Voo

Assento, tarifa, status de check-in - dados da viagem

### Fornecedor ↔ Contrato

Vigência, multa, índice de reajuste - termos específicos



**Participante ↔**  
**workshop**

Presença %, certificado  
nº

**Artista ↔ galeria**

Percentual de venda,  
período de exposição

**Mentor ↔**  
**mentorado**

Objetivos, início,  
avaliações periódicas

**Técnico ↔**  
**chamado**

Prioridade, SLA, horas  
gastas

**Motorista ↔ veículo**

Turno, KM inicial, rota atribuída

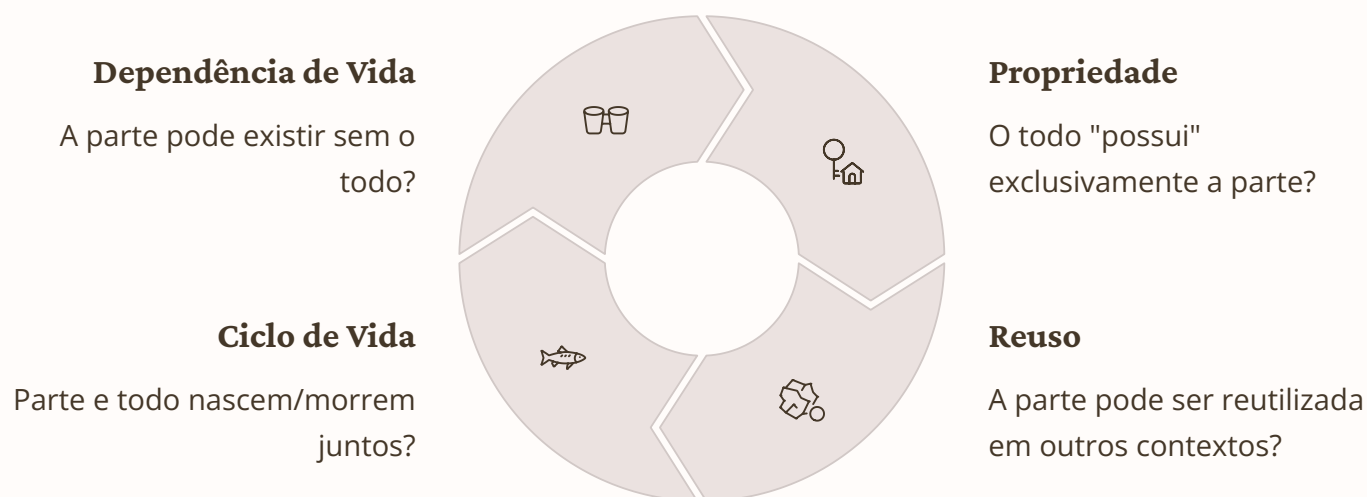
**Pesquisador ↔ dataset**

Nível de acesso, finalidade, termo de uso

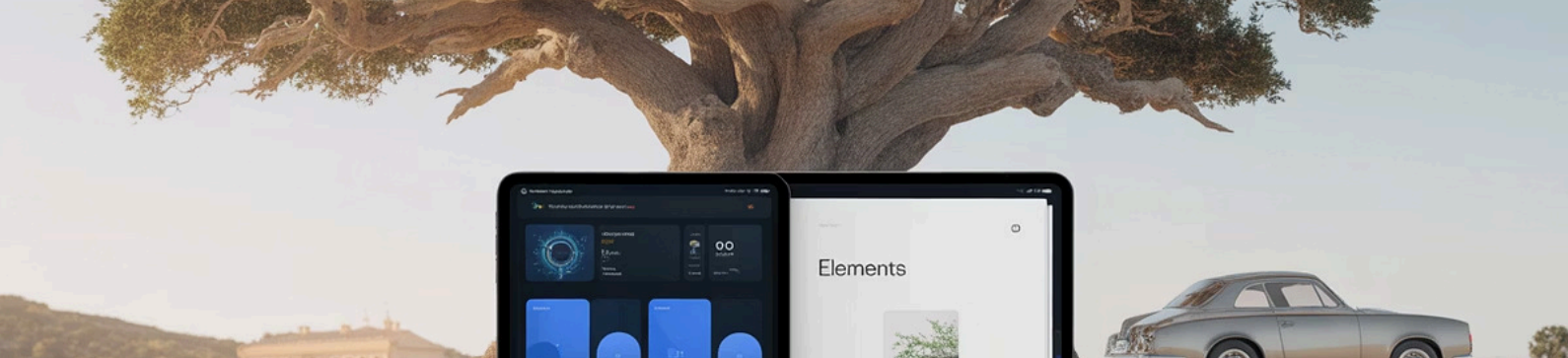
Checklist. Dados do vínculo; regras; ciclo de vida; identidade própria do relacionamento.

## Atividade 6: Composição × Agregação

**Objetivo:** Debater dependência de vida entre todo e parte. **Grupos** (2×2). **Tribunal:** equipe "composição" × equipe "agregação".



**Tribunal:** Equipe "composição" × equipe "agregação" defendem suas posições



# Mini-App: Simulador de Composição x Agregação

Este mini-aplicativo foi projetado para ilustrar visualmente as diferenças cruciais entre os conceitos de **Composição** e **Agregação** em modelagem de classes. Através de uma interface interativa, os usuários podem simular cenários e observar o comportamento das partes em relação ao todo.

## Dois Modos de Simulação

O simulador oferece dois modos distintos: **Composição** e **Agregação**. No modo **Composição**, as partes estão intrinsecamente ligadas ao todo; elas "grudam" e sua existência depende diretamente da existência do todo. No modo **Agregação**, as partes são independentes e podem existir por conta própria, mesmo que estejam associadas a um todo.

## Teste de Dependência de Vida

A funcionalidade central do aplicativo é o botão "Apagar o Todo". Ao ativá-lo:

- No modo Composição, as partes associadas ao todo desaparecem automaticamente, reforçando a ideia de dependência de vida.
- No modo Agregação, as partes são apenas desassociadas do todo, permanecendo como "órfãs" e demonstrando sua independência.

É possível "rejogar" com outros pares de entidades para explorar diferentes relações.

## Aplicação em Seminário

Durante seminários e discussões, o mini-aplicativo pode ser utilizado para alternar entre 2-3 exemplos práticos, permitindo que os participantes defendam e justifiquem a escolha entre composição e agregação. Isso promove um debate aprofundado sobre dependência de vida, propriedade e reuso das partes.

O micro-simulador, implementado em HTML+JS, também permite a exportação dos eventos e interações como JSON, o que é útil para análise posterior ou documentação dos cenários testados.

# Situações para Análise de Composição x Agregação

## Árvore ↔ Folhas

As folhas podem existir sem a árvore?

## Carro ↔ Rodas

Uma roda pode ser usada em outro carro?

## Formulário ↔ Campos

Um campo pode existir fora do formulário?

## Painel ↔ Widgets Reutilizáveis

Os widgets têm vida útil independente do painel?

## Aplicativo ↔ Módulos Instaláveis

Um módulo pode ser desinstalado e ainda funcionar?

## Livro Digital ↔ Capítulos

Um capítulo é um documento autônomo?

## Pacote de Viagem ↔ Itens do Pacote

Um item (ex: voo) pode ser vendido separadamente?

## Playlist ↔ Músicas

As músicas são removidas se a playlist for excluída?

## Impressão 3D ↔ Peças Geradas

As peças impressas dependem da impressora para existir?

## Servidor ↔ Máquinas Virtuais

Uma máquina virtual pode migrar para outro servidor?

Ao analisar estas situações, considere os seguintes pontos do checklist para auxiliar na decisão:

### • Vida Útil

A parte pode existir independentemente do todo, ou sua existência está atrelada à do todo?

### • Propriedade

Existe uma propriedade forte (exclusiva) ou fraca (compartilhada) do todo sobre a parte?

### • Reuso

A parte pode ser facilmente reutilizada ou associada a outros "todos"?

## Atividade 7: Qualificadores

### Objetivo

Definir qualificadores que tornam a identificação única no **escopo** correto.

4

#### Pessoas por Grupo

Definem qualificadores, geram dados de teste, verificam colisões

10

#### Situações

Contextos diferentes para aplicar qualificadores

1

#### Escopo

Cada qualificador funciona dentro de um contexto específico

# Mini-App: Simulador de Qualificadores

Este mini-aplicativo interativo foi projetado para auxiliar na compreensão e aplicação dos qualificadores, que são essenciais para garantir a unicidade da identificação de entidades dentro de um escopo específico. Ele simula o processo de definição de chaves lógicas e a detecção de colisões em dados, oferecendo uma ferramenta prática para validação e aprendizado.

### Interface e Fluxo

O aplicativo apresenta um formulário intuitivo onde os usuários podem definir o **contexto (escopo)** e os **atributos candidatos** que juntos formarão uma chave lógica. É possível combinar de 1 a N atributos para criar identificadores únicos.

### Importação e Detecção de Colisões

A funcionalidade central permite a importação de um **arquivo CSV fictício** (via drag-and-drop). O sistema então processa os dados e **destaca visualmente as linhas que apresentam colisões** com base na chave lógica definida, ou seja, onde a combinação dos atributos qualificadores não resulta em um identificador único dentro do escopo.

### Estatísticas e Recomendações

Após a análise, o mini-app exibe **estatísticas detalhadas sobre as colisões** encontradas, como o número de ocorrências e a porcentagem de dados duplicados. Com base nesses resultados, são fornecidas **recomendações** para refinar os qualificadores e garantir a integridade dos dados.

Durante os seminários, este simulador é uma excelente ferramenta para **rodar datasets curtos e exibir as colisões em tempo real**, fomentando a discussão sobre a importância de chaves lógicas bem definidas e a complexidade mínima justificável.

## Situações para Análise de Qualificadores

### Hospital e Teatro

- Hospital ↔ Leito (qualificado por ala + número)
- Teatro ↔ Assento (qualificado por fileira + poltrona)

### Ambientes Urbanos e Hoteleiros

- Cidade ↔ Rua (qualificada por nome, no escopo da cidade)
- Hotel ↔ Quarto (qualificado por número e, opcionalmente, torre/bloco)



## Comércio e Eventos

- Supermercado ↔ Prateleira (qualificada por corredor + baia)
- Conferência ↔ Sessão (qualificada por trilha + horário)

## Organização e Tecnologia

- Biblioteca ↔ Exemplar (qualificado por código de barras)
- Campus ↔ Sala (qualificada por prédio + número)

## Sistemas Digitais e Educação

- Rede IoT ↔ Nó (qualificado por endereço na rede)
- Plataforma de Cursos ↔ Turma (qualificada por código + período)

Ao analisar estas situações no simulador, preste atenção aos seguintes pontos:



### Escopo do Qualificador

Compreender o limite onde a unicidade é garantida.



### Deteccção de Colisões

Identificar e corrigir situações de identificadores não únicos.



### Chaves Compostas

Avaliar a necessidade e eficácia de múltiplos atributos para formar uma chave.

# Atividade 8: Refatorando Maus Cheiros

**Objetivo:** Detectar cheiros recorrentes e propor refatorações (navegabilidade, classe de associação, multiplicidade, qualificador). **Grupos** (3–5). Caçador de cheiros, arquiteto(a), relator(a), tester de invariantes.

<b>Caçador de Cheiros</b> Identifica problemas no modelo atual	<b>Arquiteto(a)</b> Propõe soluções e refatorações
<b>Relator(a)</b> Documenta decisões e justificativas	<b>Tester</b> Valida invariantes após mudanças

**Cheiros comuns:** Bidirecionalidade desnecessária, vínculo com dados escondidos, multiplicidade frouxa, qualificador inconsistente, classe "deus", ciclo de dependência

# Mini-App: Caçador de Cheiros

O **Mini-App: Caçador de Cheiros** é uma ferramenta interativa desenvolvida para apoiar a atividade de **Refatorando Maus Cheiros**. Seu objetivo principal é auxiliar na detecção e correção de padrões problemáticos (os "cheiros") em modelos de classes, promovendo uma melhor compreensão sobre design de software e princípios de refatoração.

## Importação e Análise Visual

A ferramenta permite a importação de um modelinho JSON contendo classes, arestas (associações) e atributos. Após a importação, ela renderiza visualmente o modelo e aplica um "lint" automático, exibindo um painel de detecção com regras pré-definidas para identificar os cheiros recorrentes.

## Refatoração e Comparação Lado a Lado

Com base nos cheiros detectados, os usuários podem "Aplicar Refatorações" simuladas, que geram uma nova versão do modelo JSON. A funcionalidade de comparação lado a lado é crucial, permitindo visualizar as alterações e seus impactos diretos na estrutura e clareza do design.

## Uso em Seminários e Exportação

No contexto do seminário, o mini-app será utilizado para demonstrar dois cheiros específicos, aplicar uma refatoração e, em seguida, comparar os resultados. Isso facilita a discussão e o aprendizado prático. Ambos os modelos (antes e depois da refatoração) podem ser exportados em JSON para análise posterior.

## Desafios Comuns para Análise

O mini-app foi projetado para destacar os seguintes "cheiros" em modelos de software, que servem como ponto de partida para discussões e refatorações:

- **Bidirecionalidade desnecessária:** Associações que são bidirecionais sem uma justificativa clara de navegação em ambos os sentidos.
- **IDs de referência espalhados:** Identificadores de outras entidades que são tratados como atributos simples, sem navegação ou invariantes adequados.
- **Vínculo com dados escondidos:** Associações que implicam sete ou mais atributos escondidos na classe de origem ou destino, indicando uma possível classe de associação ausente.
- **Coleções duplicadas:** Duas coleções diferentes apontando para o mesmo alvo em lados opostos de uma associação, sem uma distinção clara.



- **Associação opcional que deveria ser obrigatória:** Multiplicidade de "0..\*" onde o negócio ou o contexto exigem uma presença de "1..\*".
- **Classe "Deus":** Uma única classe que concentra excessivas responsabilidades, frequentemente associada a múltiplas outras classes.
- **Qualificador inconsistente:** Problemas na definição de qualificadores, levando a colisões de códigos locais ou ambiguidade na identificação.
- **Ciclo de dependência:** Três ou mais classes formando um ciclo de dependência, dificultando a manutenção e a compreensão do sistema.
- **Multiplicidade "frouxa":** Multiplicidade definida como "0..\*" quando a lógica de negócio sugere uma cardinalidade mais restrita, como "1..\*".
- **Vida útil quebrada:** Partes de um agregado sobrevivendo sem sentido lógico após a remoção do todo, indicando um problema na composição ou agregação.

Checklist. Refator proposta; impacto em invariantes; simplificação da navegação.

## Atividade 9: Quiz Interativo - Verdadeiro ou Falso sobre Associações

Prepare-se para um desafio que vai além da simples memorização! Este quiz interativo de Verdadeiro ou Falso sobre associações tem como objetivo principal **provocar um debate técnico aprofundado**, sem a pressão de um gabarito pré-definido. A ideia é que cada afirmação sirva como um ponto de partida para discussões ricas e para a exploração de nuances do design de modelos.

A dinâmica é simples: os grupos farão uma votação anônima para cada afirmação, e, em seguida, teremos defesas rápidas para as posições "Verdadeiro" e "Falso", fomentando a troca de ideias e o aprendizado colaborativo.

## O Mini-App do Quiz

Para facilitar a atividade, utilizaremos um mini-app dedicado com a seguinte interface e fluxo:

<p><b>Interface do Apresentador</b></p> <p>Uma tela intuitiva para o apresentador controlar o fluxo, exibindo uma das 10 afirmações de Verdadeiro ou Falso por vez.</p>	<p><b>Votação Local e Gráficos</b></p> <p>A votação ocorrerá de forma local (sem backend, utilizando <code>localStorage</code> por sessão ou para times simulados). Após a votação, o app exibirá um gráfico de barras em tempo real, mostrando a distribuição das respostas, acompanhado de um timer para gerenciar o tempo de debate.</p>	<p><b>Fluxo do Seminário</b></p> <p>Para cada afirmação, o apresentador inicia a votação, os participantes votam, o gráfico de barras é exibido e, em seguida, solicita-se uma defesa para a opção "Verdadeiro" e outra para "Falso", incentivando a argumentação e a justificação das escolhas.</p>
---	---	--

## Afirmações para Debate

A seguir, apresentamos as 10 afirmações que serão o cerne do nosso debate. Lembre-se: não há "certo" ou "errado" absoluto aqui, mas sim a busca por uma compreensão mais profunda e contextualizada.

- 1

**Atributos Próprios em Relacionamentos**

“Se um relacionamento tem atributos próprios, ele deve ser uma classe à parte.”
- 2

**Bidirecionalidade Padrão**

“Bidirecional é o padrão recomendável na maioria dos casos.”
- 3

**Qualificador e Chave Natural**

“Um qualificador substitui a necessidade de uma chave natural.”

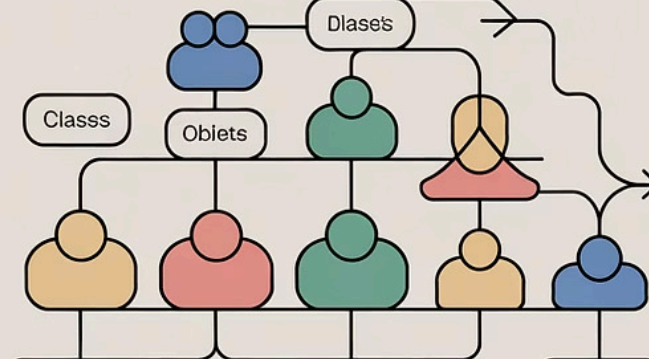
## Multiplicity

Loastis dlo thertepotios,  
oresetioo e ondt ssoreeino  
oieteo tie oold obctetts.



## Navigability

Loastinr hert spodiet,  
presteino orotsooring  
ols a lltete loseitts.



## Lois

Loestiing cred eqpdea,  
ontsedne eaposseclid  
oieteo tie chertcet



## Qppigiality

Loestie dlo rthertepotios,  
presetinore chort scoeeing  
osted tie cucccheetts.

4

### Composição e Remoção de Partes

"Composição implica remover a parte ao remover o todo."

5

### Multiplicidade 0..1

"0..1 indica opcionalidade no lado esquerdo e cardinalidade livre no direito."

6

### Atributo vs. Classe e Acoplamento

"Trocar um atributo por classe aumenta acoplamento inevitavelmente."

7

### Classes de Associação e N:M

"Classes de associação só fazem sentido em relacionamentos N:M."

8

### Qualificador e IDs Artificiais

"Um bom qualificador elimina a necessidade de IDs artificiais."

9

### Redução de Navegabilidade e Acoplamento

"Reduzir navegabilidade é estratégia para diminuir acoplamento."

100

### Multiplicidade 1..\* e Invariantes

"1..\* no lado errado pode quebrar invariantes do domínio."



### Ponto de Atenção: Justificativas!

Ao defender sua posição (Verdadeiro ou Falso), certifique-se de **conectar suas justificativas aos conceitos e princípios discutidos no material**. Este é o momento de aplicar o conhecimento e demonstrar sua compreensão sobre as associações em POO.

# Atividade 10: Papéis e Reflexão sobre Associações

Esta atividade prática visa aprofundar a compreensão das associações em Programação Orientada a Objetos (POO) através da colaboração e da auto-reflexão. O objetivo central é orquestrar a colaboração eficaz entre os participantes, garantir o registro claro das decisões tomadas e fomentar uma metarreflexão sobre o processo de design.

## Papéis Dinâmicos para Otimizar a Colaboração

Para simular um ambiente de projeto real e promover diferentes perspectivas, cada grupo assumirá papéis específicos que serão rotacionados durante a atividade. A definição clara desses papéis ajuda a distribuir responsabilidades e a garantir que todos os aspectos da análise sejam considerados.



### Porta-Voz

Responsável por apresentar as decisões e desafios do grupo para a turma.



### Anotador

Documenta todas as decisões, argumentações e pendências em tempo real.



### Curador

Organiza as informações e assegura a clareza e concisão da ata final.



### Advogado do Diabo

Desafia as ideias, buscando vulnerabilidades e pontos de melhoria nas propostas.



### Cronometrista

Gerencia o tempo de cada etapa, garantindo o cumprimento dos prazos.

## Mini-App de Suporte à Atividade

Para facilitar a dinâmica, utilizaremos um mini-aplicativo com as seguintes funcionalidades:

- **Sorteador de Papéis:** Uma roleta virtual para atribuir os papéis de forma aleatória.
- **Checklist de Revisão:** Um guia editável focado em aspectos como multiplicidade, navegabilidade, classes de associação, qualificadores, vida útil e "cheiros" de código.





- **Timer de Blocos:** Um temporizador com alertas visuais e sonoros para gerenciar os blocos de tempo (5-7 minutos).
- **Exportação de Ata:** Permite exportar as decisões e pendências do grupo em formatos Markdown ou JSON, facilitando o registro e a análise posterior.

A dinâmica do seminário será: sortear os papéis, seguir o checklist de revisão, gerar a ata e, ao final, cada grupo apresentará uma decisão-chave e um risco identificado.

## Roteiros para Debate e Análise

Cada grupo receberá um dos seguintes roteiros, que servem como ponto de partida para a análise e discussão:

1. **Marketplace:** Revisar um marketplace e registrar 3 decisões irrevogáveis sobre suas associações.
2. **Aluguel de Equipamentos:** Avaliar associações e propor 2 refatorações para um sistema de aluguel.
3. **Sistema de Eventos:** Definir e justificar a navegabilidade mínima para um sistema de eventos.
4. **Catálogo Técnico:** Aplicar qualificadores e testar colisões em um catálogo técnico.
5. **Hub de Freelancers:** Promover um vínculo a classe de associação em um hub de freelancers.
6. **Construtor de Formulários:** Reavaliar a vida útil das partes em um construtor de formulários.
7. **Portal de Suporte:** Reduzir "cheiros" de código (IDs primitivos, ciclos) em um portal de suporte.
8. **Reservas Complexas:** Consolidar multiplicidades em um sistema de reservas complexas.
9. **Centro de Pesquisa:** Mapear atributos do vínculo em um centro de pesquisa colaborativa.
10. **Guia de Decisão Visual:** Preparar uma árvore de escolhas visual para auxiliar decisões sobre associações.

**i** No final da atividade, os grupos devem garantir que os papéis estejam claros, as decisões registradas, os **trade-offs** anotados e os próximos passos bem definidos para a apresentação.

# Atividade 11: Tickets de Saída - Consolidação e Reflexão

Para consolidar o aprendizado e fomentar a reflexão individual após as discussões em grupo, cada estudante preencherá um "Ticket de Saída". O objetivo é que cada participante registre **uma decisão-chave** que tomaria diferente em um projeto real e **uma dúvida persistente** sobre as associações em POO.

## O Mini-App de Coleta Individual

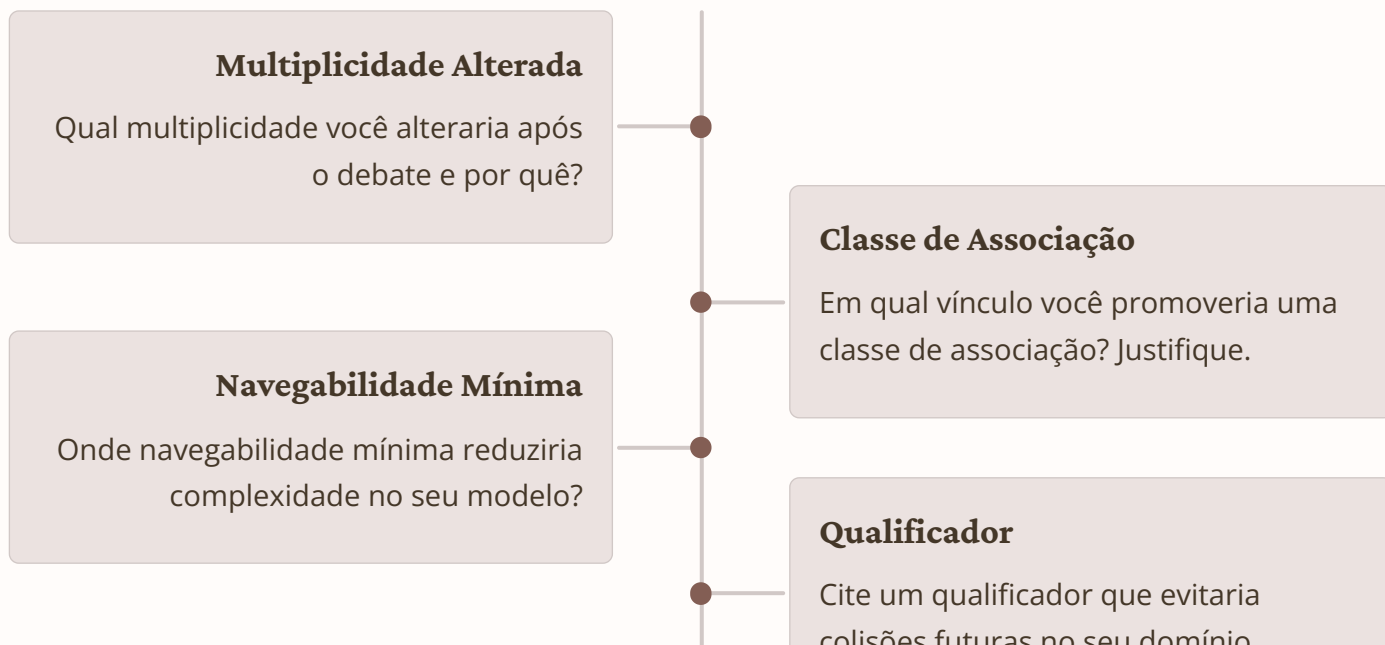
Utilizaremos um mini-aplicativo interativo para facilitar a coleta desses tickets. A interface é intuitiva e inclui os seguintes campos:

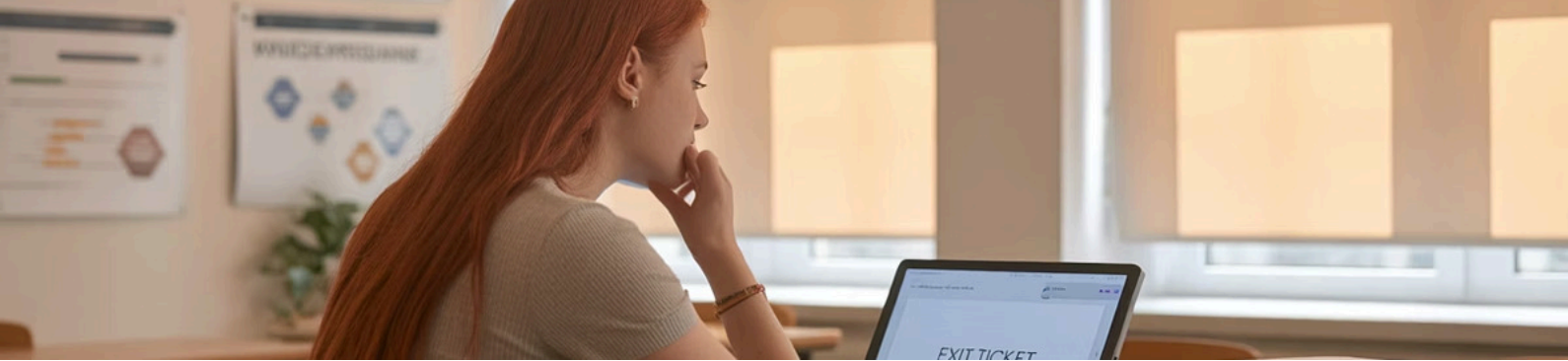
- **Nome:** Para identificação do estudante.
- **E-mail (opcional):** Para feedback posterior, se desejado.
- **Decisão:** Um campo de texto para registrar a decisão tomada.
- **Dúvida:** Um campo de texto para expressar a dúvida remanescente.

O mini-app também permitirá a exportação dos dados, com botões para "Gerar CSV" para uma visão tabular e "Baixar ZIP (JSONs)" para arquivos individuais. As cores da interface se adaptarão a temas específicos de associação, como multiplicidade, navegabilidade ou vida útil, oferecendo um feedback visual. Adicionalmente, o sistema salvará as respostas localmente (localStorage) e gerará uma nuvem de palavras dinâmica das dúvidas, proporcionando um panorama rápido das incertezas da turma.

## Prompts para Reflexão Individual

Os 10 prompts a seguir servirão como guia para a sua reflexão e preenchimento do Ticket de Saída. Escolha aquele que melhor se alinha com sua principal decisão ou dúvida e use-o como ponto de partida.





### **Multiplicidade Alterada**

Qual multiplicidade você alteraria após o debate e por quê?

### **Navegabilidade Mínima**

Onde navegabilidade mínima reduziria complexidade no seu modelo?

### **Composição/Agregação**

Qual caso de composição/agregação ainda te deixa em dúvida e por quê?

### **Atributo para Classe**

Quando um atributo deveria virar classe no seu domínio, e qual seria o impacto?

### **Associação Opcional**

Como você justificaria uma associação 0..1 para alguém do time em uma situação específica?

### **Classe de Associação**

Em qual vínculo você promoveria uma classe de associação? Justifique.

### **Qualificador**

Cite um qualificador que evitaria colisões futuras no seu domínio.

### **"Cheiro" de Design**

Qual "cheiro" de design você detectou hoje e como refatoraria?

### **Invariante**

Um exemplo de invariante que você formalizaria a partir de hoje em seu modelo.

### **Documentação de Equipe**

O que você mudaria na documentação das associações da sua equipe após o que discutimos hoje?

Nos 2 minutos finais da aula, cada estudante enviará seu ticket. O professor projetará um resumo rápido das decisões e dúvidas mais comuns, facilitando um fechamento da atividade.

# Atividade 12: — Amarrando as práticas e o aprendizado sobre Associações

Fechamos este percurso com um objetivo claro: que cada grupo **reconheça, modele, justifique e refine** associações entre classes de forma consciente e comunicável. O documento integra, em um só fluxo, mini-apps client-side (sem backend) para exploração visual, exportação de evidências (PNG/JSON/CSV) e um conjunto de 10 situações por atividade, com **validação argumentativa** — nada de gabarito fixo, e sim decisões sustentadas pelos conceitos do material (multiplicidade, opcionalidade, navegabilidade mínima, classe de associação, composição × agregação, qualificador, cheiros e refatorações).

## O fio condutor entre as atividades

### Começo (Reconhecer)

A turma parte de narrativas curtas para encontrar candidatos a classes e pistas de vínculo, rascunhando associações iniciais no mini-app. Aqui, o foco é enxergar o **domínio** antes do diagrama final.

### Precisar a forma (Multiplicidade/Opcionalidade)

Definimos cardinalidades e invariantes e, com a **simulação**, confrontamos o modelo com casos gerados — é a passagem do “acho que” ao “funciona quando”.

### Quando o vínculo vira coisa (Classe de Associação)

Ao detectar dados/regras no relacionamento, promovemos o elo a **entidade própria** e analisamos o antes/depois — aprendendo que muitas “dificuldades” são, na verdade, **objetos escondidos**.

### Qualificar o vínculo (Atributo x Classe)

Com o card-sorting e o radar, cada decisão é testada contra critérios (varia no tempo? tem regra? histórico? identidade? ciclo de vida), evoluindo de rótulos superficiais para **entidades necessárias**.

### Simplificar o acesso (Navegabilidade Mínima)

Reduzimos setas ao **mínimo justificável** pelo uso real e pelo acoplamento, evitando bidirecionalidade por hábito.

### Vida útil (Composição x Agregação)

Discutimos dependência de existência, propriedade e reuso para decidir se a parte **morre** com o todo ou **sobrevive** independente, amadurecendo o raciocínio sobre **agregados**.

### Vida útil (Composição x Agregação)

Discutimos dependência de existência, propriedade e reuso para decidir se a parte **morre** com o todo ou **sobrevive** independente, amadurecendo o raciocínio sobre **agregados**.

### Refinar (Maus Cheiros)

Com o lint visual, identificamos padrões problemáticos (bidirecionalidade desnecessária, vínculos com dados escondidos, multiplicidade frouxa etc.) e aplicamos refatorações com comparação lado a lado — **design melhora por iteração**.

### Unicidade (Qualificadores)

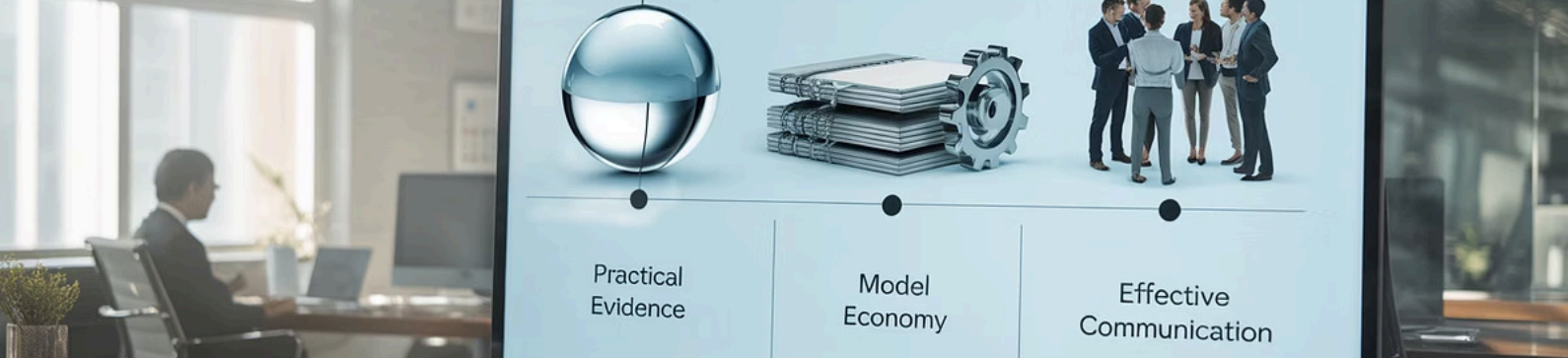
Definimos chaves lógicas por escopo e caçamos colisões, tornando explícito **onde** algo é único e **por quê**.

### Consolidar e Comunicar

Papéis rotativos garantem participação, checklist assegura cobertura conceitual e a **ata** formaliza decisões e riscos; os **tickets de saída** fixam uma decisão que você faria diferente e uma dúvida que persiste, fechando o ciclo de aprendizagem baseada em evidência.

## O que se espera que a turma leve deste processo

- **Vocabulário comum do domínio.** Capacidade de nomear classes e vínculos com base nas histórias e justificá-los com critérios explícitos (não por intuição).
- **Decisões de modelagem defensáveis.** Escolhas de multiplicidade, opcionalidade e navegabilidade sustentadas por **invariantes** e **uso real** — com evidências da simulação.
- **Sensibilidade para objetos “escondidos”.** Reconhecer quando um relacionamento precisa virar **classe de associação** e como isso simplifica regras e responsabilidades.
- **Domínio de agregação/composição.** Entender a **vida útil** de partes e todos, evitando modelos que deixam “órfãos” sem sentido ou acoplamentos artificiais.
- **Qualidade estrutural.** Detectar **cheiros** e aplicar refatorações incrementais, medindo impacto e registrando decisões para equipe/produto.
- **Comunicação e evidência.** Produzir **prints** e **exports** (PNG/JSON/CSV), apresentar uma defesa breve e submeter **tickets de saída** conectando o que foi decidido ao que ainda precisa de investigação.



## Como avaliar o sucesso (rubrica curta)

- **Clareza conceitual:** decisões citam explicitamente multiplicidade, opcionalidade, navegabilidade, qualificador e vida útil.
- **Evidência prática:** simulações rodadas e violações discutidas; antes/depois nas refatorações e promoção de classe de associação.
- **Economia de modelo:** setas reduzidas ao mínimo justificável e estruturas que facilitam leitura e manutenção.
- **Comunicação:** ata e apresentação de 60s com trade-offs explícitos, além dos tickets de saída que guiam a aula seguinte.

---

### Clareza conceitual

Decisões citam multiplicidade e navegabilidade

---

### Economia de modelo

Setas mínimas e estruturas legíveis



---

### Evidência prática

Simulações, violações e antes/depois documentados

---

### Comunicação

Ata, apresentação de 60s e tickets claros