

E-book Módulo 2 - Descobrindo como Usar Aggregates, Modelagem de Dados e Índices

- O que é um Aggregate?
- Criando um Aggregate a partir do uso do Componente List
- Adicionando ação de Exclusão de Registro em um ListItem
- Desenvolvendo uma Ação para Criar Novos Registros
- O que é Modelagem de Dados?
- Tipos de Relacionamentos entre Entidades de Banco de Dados
- Conhecendo Índices
- Encerramento

O que é um Aggregate?

O que é um Aggregate?

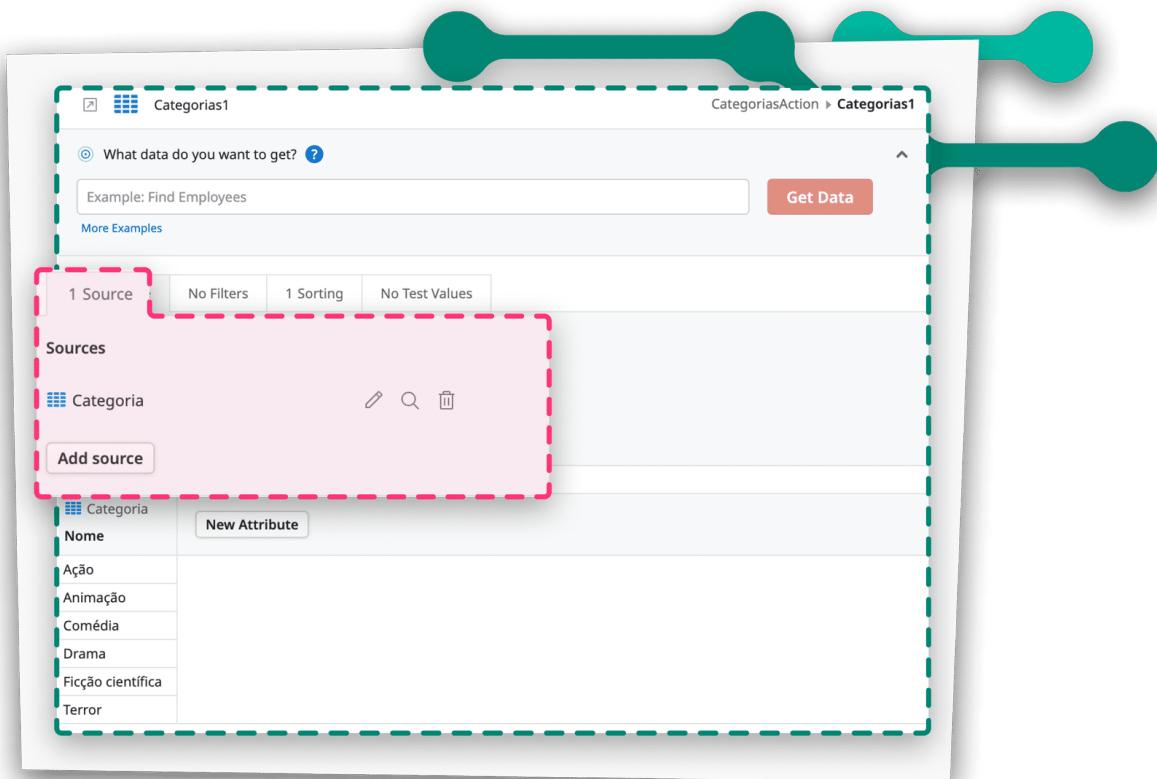
"Aggregate é um elemento visual que permite criar uma consulta voltada a buscar dados a partir de entidades aplicando critérios de filtro e ordenação."

Esse elemento pode ser usado por **Widgets**, ou melhor, componentes visuais, como por exemplo, o componente **List**.

Outra possibilidade de uso é em **ações**. Dessa forma, quando uma ação precisar recuperar dados de uma ou mais entidades, Aggregate poderá ajudar a acessar esses dados desejados durante a execução da ação.

Conhecendo no detalhe um Aggregate

Em um Aggregate são oferecidas quatro abas que permitem definir os critérios para recuperar os dados desejados. As abas são as seguintes: **Source**, **Filters**, **Sorting** e **Test Values**. Abaixo, cada aba é apresentada.



Source

Essa aba permite indicar quais entidades são as fontes de dados desse Aggregate.

Neste exemplo, perceba que está sendo usado uma entidade chamada Categoria.

Filters

Nesta aba podem ser criados filtros que ajudem a limitar os dados que serão recuperados por algum Aggregate.

Perceba que na imagem ao lado há dois Filtros colocados.

1) Length(Categoria.Nome) > 6: Recupera os registros que tenham no atributo **Nome** mais do que **6 caracteres**.

2) Categoria.Nome like "%ma%": Recupera nomes que tenham "**ma**" em algum local do nome da categoria correspondente.

Caso o desenvolvedor deseje adicionar mais filtros, poderá criar quantos achar necessário.

The screenshot shows a user interface for querying data. At the top, there's a search bar with the placeholder "Example: Find Employees" and a "Get Data" button. Below the search bar are four status indicators: "1 Source", "2 Filters", "1 Sorting", and "No Test Values". A pink dashed box highlights the "2 Filters" indicator and the filter configuration below it. The filters are defined as:

- 1: Length(Categoria.Nome) > 6
- 2: Categoria.Nome like "%ma%"

Below the filters is a table with columns "Categoria" and "Nome". A "New Attribute" button is visible. A teal connector is attached to the bottom right corner of the interface.

The screenshot shows a user interface for querying data. At the top, there's a search bar with the placeholder "Example: Find Employees" and a "Get Data" button. Below the search bar are four status indicators: "1 Source", "No Filters", "1 Sorting", and "No Test Values". A pink dashed box highlights the "1 Sorting" indicator and the sort configuration below it. The sort is defined as:

- 1: Categoria.Id (Descending)

Below the sort is a table with columns "Categoria" and "Nome". The "Categoria" column has headers "Id" and "Nome". A "New Attribute" button is visible. A teal connector is attached to the bottom right corner of the interface.

Sorting

Nesta aba é possível ordenar os dados a partir de algum critério escolhido.

Caso deseje, por exemplo, colocar em ordem alfabética os dados de algum atributo específico, esse é o local certo para fazer isso!

Se também quiser usar mais de um atributo para definir algum critério de ordenação, também é possível.

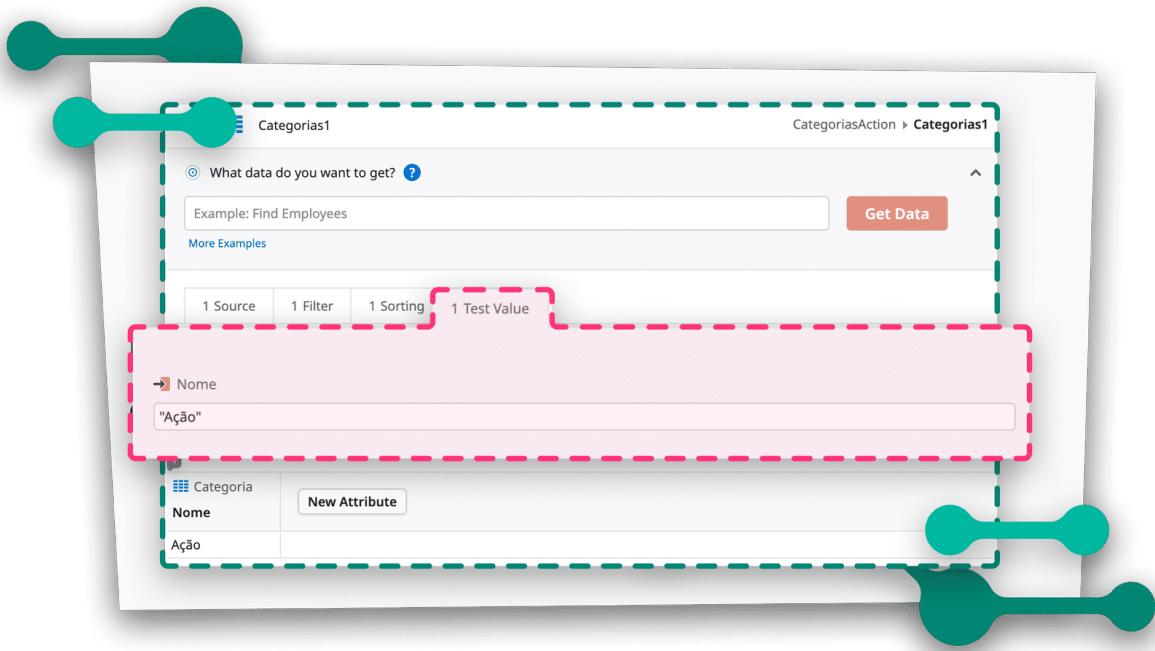
No exemplo apresentado na imagem, perceba que a ordenação está usando como base o atributo **Id** da entidade **Categoria**.

Além disso, está sendo apresentado o dado do maior para o menor, já que está ordenado de forma decrescente (*Descending*).

Test Value

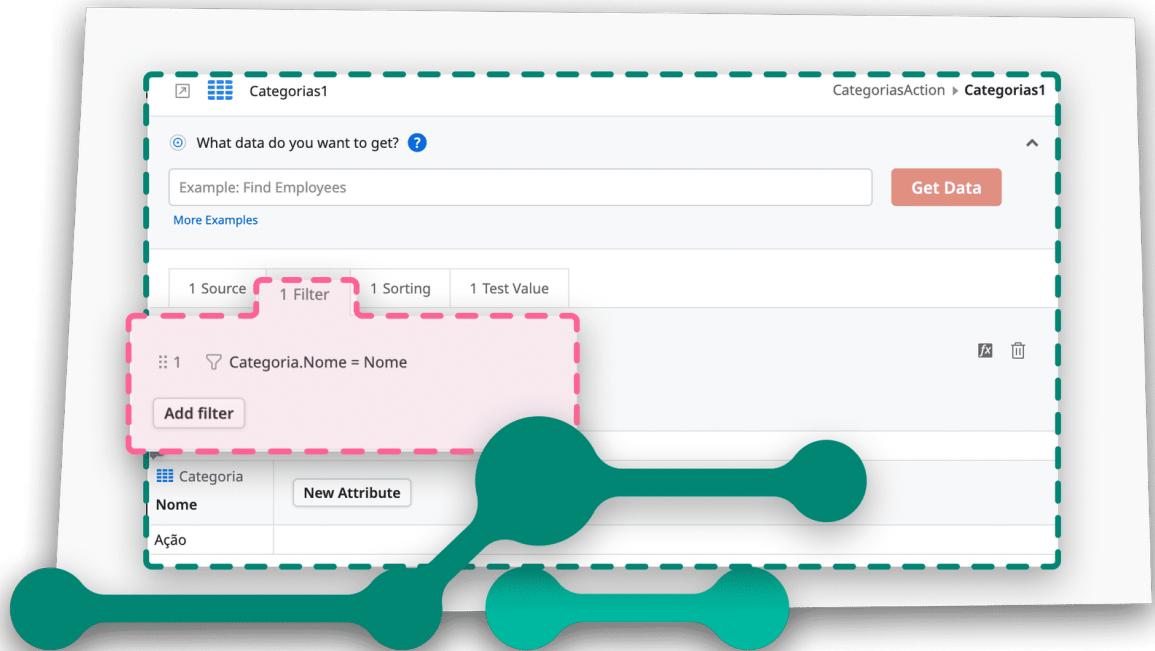
Aba que permite definir valores em variáveis externas e que poderão ser usadas em filtros ou em critérios de ordenações em algum Aggregate.

O foco dessa aba é ajudar a testar situações de recuperação de dados. Perceba que no exemplo ilustrado, há uma variável chamada **Nome** e o conteúdo fornecido nela é **Ação**.



Essa variável poderá ser usada em alguma condição incluída na aba de filtros, como está sendo mostrado ao lado.

Perceba que o filtro compara o atributo **Nome** da entidade **Categoria** com a variável que também é chamada **Nome**.



O que é SQL?

SQL significa *Structured Query Language*, ou seja, é uma linguagem de programação conhecida que é usada para realizar certas ações em bancos de dados, como por exemplo, inserção, exclusão, alteração e consulta de dados.

Visando facilitar, por exemplo, a definição dos critérios a serem usados para recuperar dados, a plataforma Low-Code da OutSystems oferece a possibilidade de uso dos Aggregates, que permitem a customização fácil e rápida desses critérios.

Adicionalmente, a plataforma integra seu uso com SQL, permitindo que caso algum desenvolvedor deseje criar alguma consulta usando a linguagem, ele poderá fazer.

Criando um Aggregate a partir do uso do Componente List

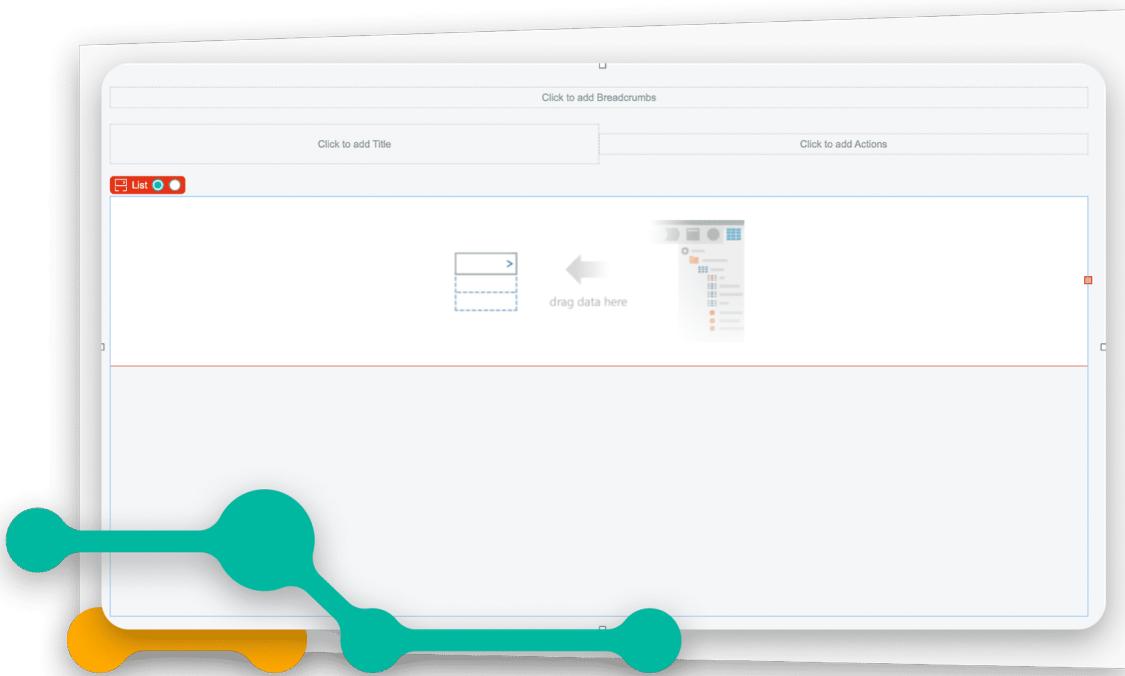
Criando um Aggregate a partir do uso do Componente List

Para usar um Aggregate a partir do componente visual List, abaixo há um conjunto de passos que orientam como pode ser feito esse uso.

Passo 1

Crie uma nova tela em projeto criado na plataforma do Service Studio oferecido pela OutSystems.

Passo 2

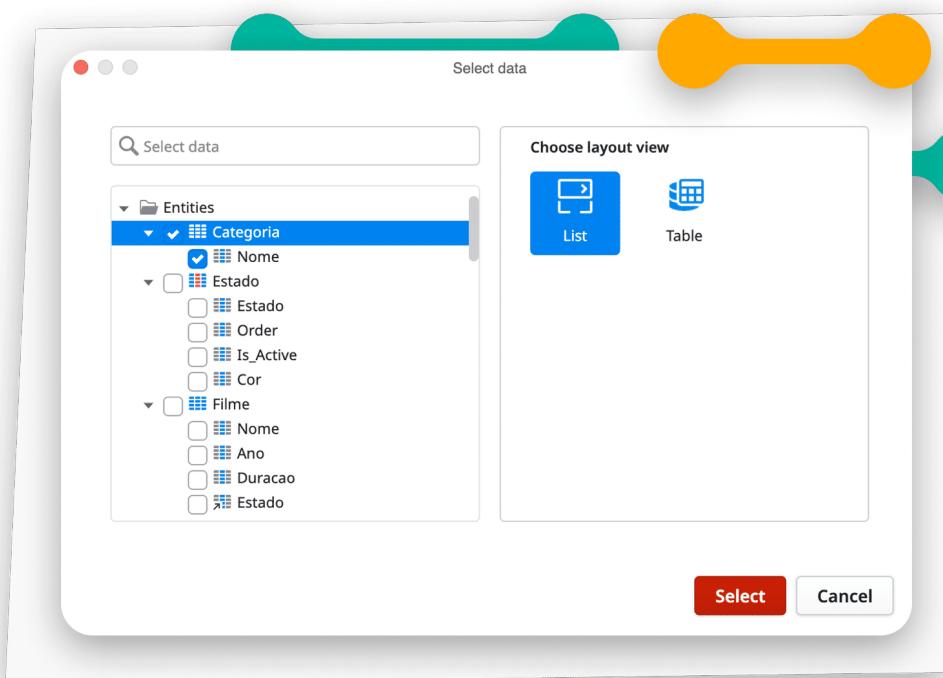


Adicione o componente List na nova tela, assim como ilustrado na imagem acima.

Passo 3

Clique duas vezes no componente List adicionado na tela.

Passo 4

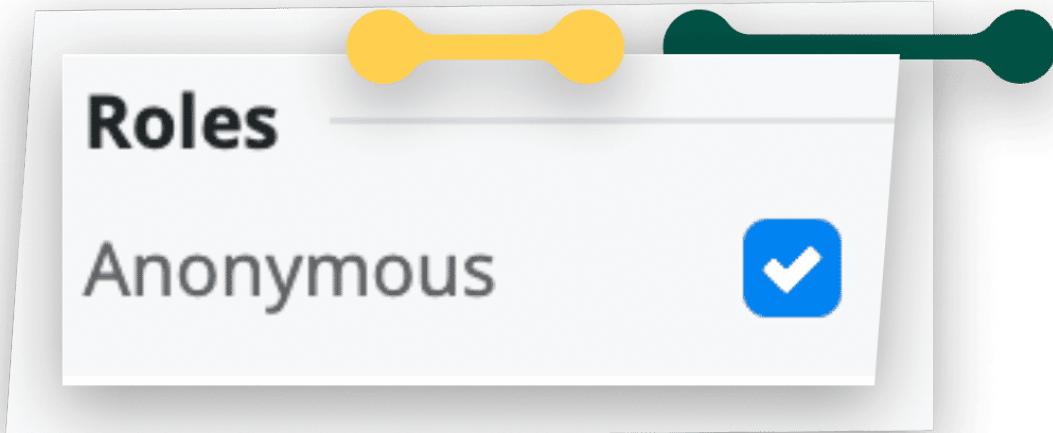


Escolha os atributos que deseja apresentar informações de registros salvos em alguma entidade. Além disso, escolha a forma como serão apresentados os dados: **List** ou **Table**.

A primeira opção apresenta os dados em um formato de lista, enquanto que a segunda opção em um formato de tabela.

Ao realizar as escolhas desejadas, clique no botão **Select**.

Passo 5



Caso não tenha um controle de acesso com usuário desenvolvido, coloque como anônima a tela criada.

Passo 6

Publique o projeto, e pronto!

Teste a tela desenvolvida!!

Adicionando ação de Exclusão de Registro em um ListItem

Adicionando ação de Exclusão de Registro em um ListItem

Abaixo há um passo a passo que explica como é possível criar uma ação que seja capaz de excluir algum registro presente em um entidade de banco dados.

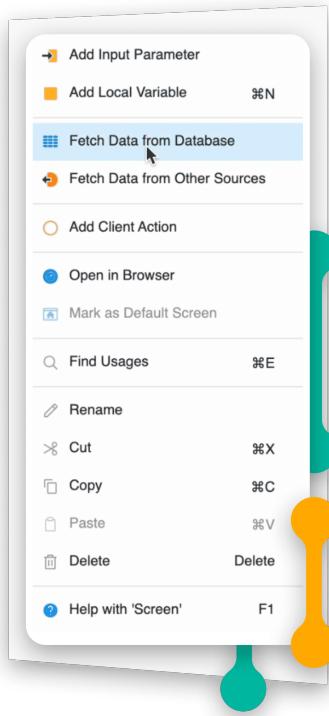
Mas como criá-la?



Passo 1

Crie uma nova tela em um projeto do Service Studio e adicione o componente ***List*** nessa tela.

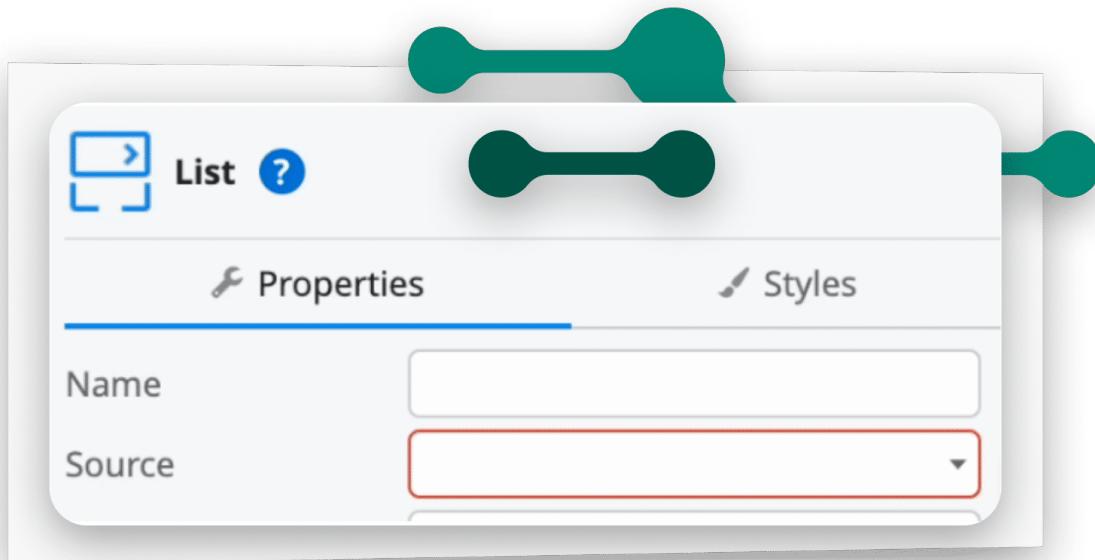
Passo 2



Crie manualmente um novo **Aggregate**, clicando com o botão direito do mouse na nova tela criada, localizada na aba Interface, e escolha a opção “**Fetch Data from Database**”, como ilustrado na imagem.

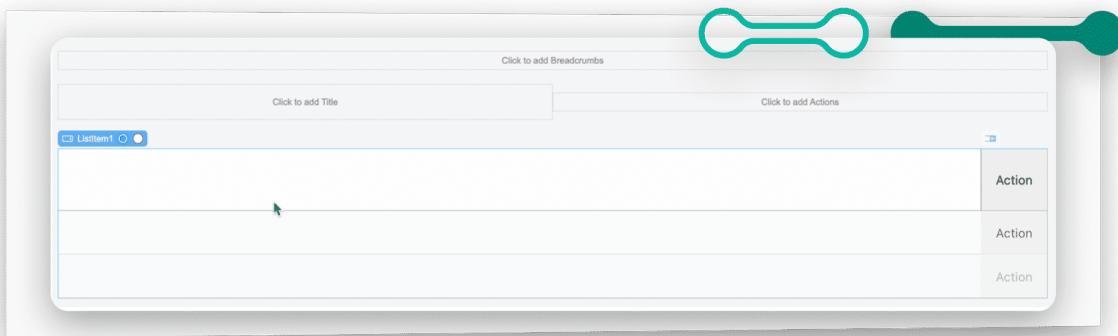
Forneça um nome para o novo **Aggregate** e defina em cada uma das abas oferecidas, as informações necessárias, como por exemplo, qual entidade de banco de dados terá seus dados recuperados na **aba Source**, assim como quais critérios de filtros e ordenações serão definidos, nas abas **Filters** e **Sorting**, respectivamente.

Passo 3



Selecione o componente **List**, que foi adicionado na tela, e indique na propriedade **Source**, que será usado o novo **Aggregate** criado.

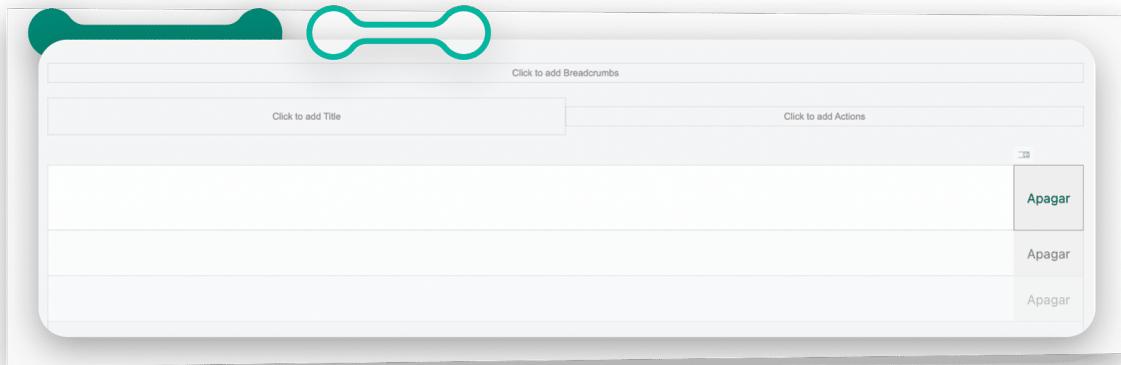
Passo 4



Arraste um componente **ListItem** para o componente **List** colocado na tela.

Você deve chegar em uma situação parecida com a ilustrada na imagem.

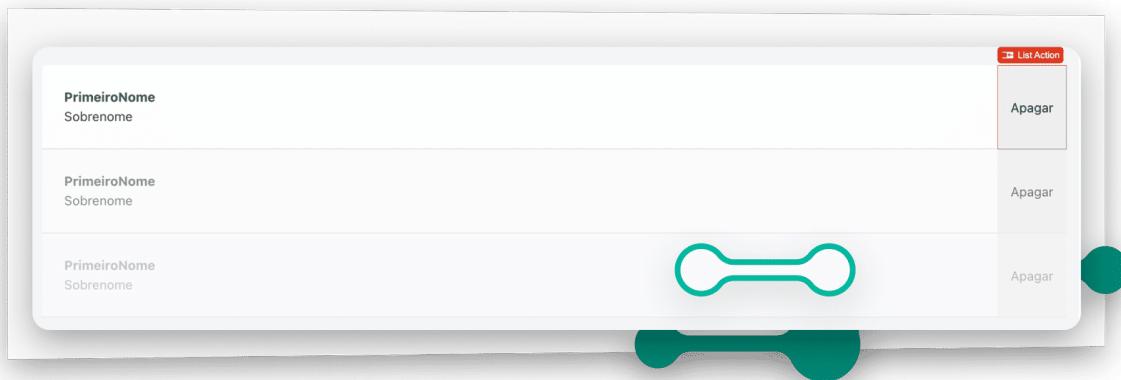
Passo 5



Mude o nome do texto “**Action**” para “**Apagar**”, “**Deletar**”, **Excluir**, ou algo similar, acessando o lado direito de cada item.

Veja um exemplo na imagem apresentada.

Passo 6



Indique quais são os atributos que serão apresentados em cada linha da lista da tela.

Para fazer isso, você pode arrastar o atributo da entidade usada pelo **Aggregate** e arrastar para a primeira linha da lista da tela.

No exemplo abaixo, duas informações serão apresentadas: primeiro nome e sobrenome de cada registro recuperado.

Passo 7



Crie uma nova ação que seja capaz de excluir algum registro. Essa ação deve estar vinculada com o botão “**Apagar**”, de cada item apresentado no componente **List**.

Para criar tal ação, o desenvolvedor poderá usar a “**Entity Action**” chamada “**Delete<NomeEntidade>**”, que é oferecida por alguma entidade já criada. Assim que incluir essa ação, procure atualizar os dados que estão sendo apresentados na tela correspondente.

Para isso, use o componente “**Refresh Data**”. Você deve criar uma ação seguindo um fluxo similar ao ilustrado abaixo, que procura deletar um registro de uma entidade **Pessoa**.

Passo 8

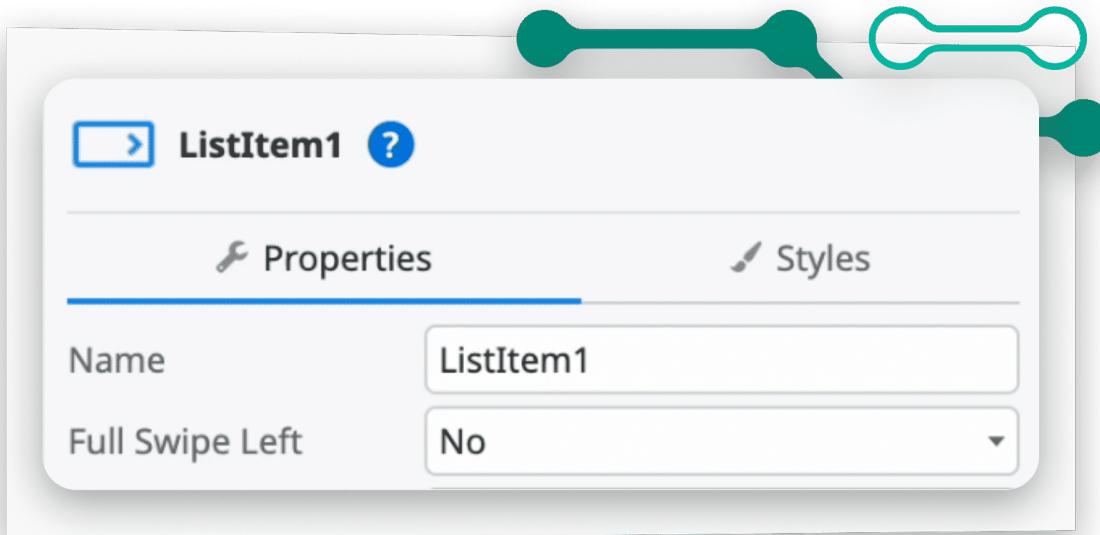
Coloque a tela criada como anônima, caso não tenha um controle de acesso com usuário desenvolvido.

Publique o projeto!

Na sequência, abra o browser, clique com o botão direito em qualquer área que a tela desenvolvida está sendo apresentada, e escolha a opção **Inspecionar**.

Para visualizar a tela no formato de um dispositivo móvel (p.ex. smartphone), selecione a visão desejada a partir do browser.

Passo 9



Faça o **Swipe** no item que deseja excluir, ou seja, arraste da direita para esquerda o Item escolhido. Perceba que ele será excluído, caso arraste até o final do lado esquerdo do item.

Caso deseje forçar o usuário a excluir o item somente clicando no botão “**Apagar**”, volte ao Service Studio e selecione um item na interface que está sendo desenvolvida.

Acesse a propriedade “**Full Swipe Left**” e selecione a opção **Não (No)**, assim como ilustrado na imagem.

Passo 10

Publique novamente o projeto e abra o browser para testar.

Agora você deve ser capaz de deletar cada item da lista somente clicando no botão apagar.

Desenvolvendo uma Ação para Criar Novos Registros

Desenvolvendo uma Ação para Criar Novos Registros

Para criar uma ação que permita a criação de novos registros, você poderá usar alguma "Entity Action" já oferecida pela entidade correspondente. Geralmente, essa "Entity Action" possui o nome "Create<NomeEntidade>".

Considerando esse cenário, vale destacar que quando é realizado o desenvolvimento de ações, há a possibilidade de criar ações no lado do cliente ou no lado do servidor.

Abaixo, é explicada a diferença de cada tipo de ação.

1- Ação Cliente

Ação executada do lado do dispositivo que o usuário está acessando o sistema.

2- Ação Servidor

Ação executada no servidor. Ela tem relação com alguma lógica do lado do servidor, manipulação de dados, lógica de negócios e operações de banco de dados.

Considerando a explicação acima, para qualquer manipulação de alteração direta de dados em alguma entidade, sugere-se que seja criada e usada alguma ação que esteja do lado do servidor.

Assim, o desenvolvedor se quiser, pode criar uma ação no lado do cliente que chame uma outra ação do servidor que ai sim irá criar, atualizar ou quem sabe excluir algum registro.

O que é Modelagem de Dados?

O que é Modelagem de Dados?

É um processo que permite criar representações visuais de dados presentes em sistemas.

Esses modelos ajudam a entender quais dados estarão presentes, assim como a forma que serão estruturados. Para melhor entender como esses modelos podem ser representados, torna-se importante saber que há três tipos principais de modelos de dados: modelo conceitual, modelo lógico, e modelo físico. **Vamos conhecê-los?**

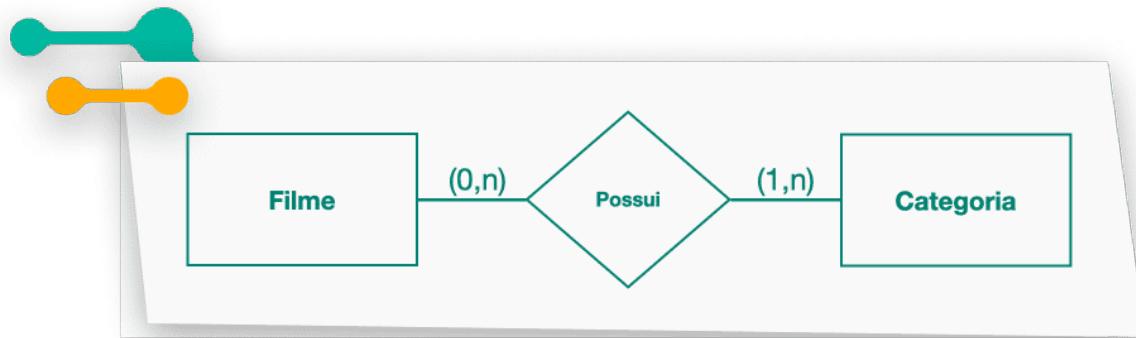
Modelo Conceitual

O modelo conceitual, também é conhecido como modelo de domínio.

Esse tipo de modelo tem como foco principal oferecer uma visão geral dos dados, mas trazendo a explicação de como eles serão organizados e quais regras de negócios devem ser consideradas. Geralmente esses modelos são criados durante a identificação e definição inicial dos requisitos que irão fazer parte de algum projeto.

Adicionalmente, esses modelos também ilustram os relacionamentos possíveis entre dados, além de informar possíveis requisitos de segurança que devem ser considerados.

Para melhor materializar um modelo conceitual, veja o modelo apresentado na imagem.

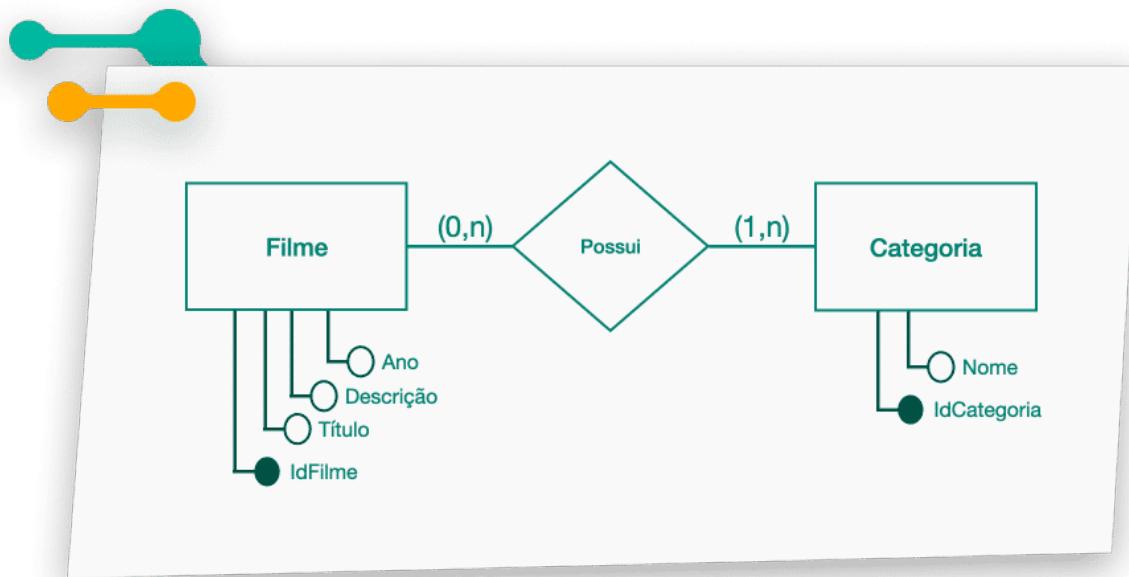


Nesse modelo é informado que **Filme** possui **Categoria**, ou vice versa. Para informar a quantidade mínima e máxima de cada lado envolvido, podem ser incluídas **cardinalidades**, que são representadas aqui por **(0,n)** e **(1,n)**. A partir delas é possível entender que cada Filme tem que estar relacionado com no mínimo uma categoria até muitas delas, que é

representado pela letra n. Já cada Categoria pode ter nenhum, assim como vários filmes relacionados com ela.

Caso deseje modelar quais dados deverão estar presentes em cada entidade, uma possibilidade é modelar assim como está sendo ilustrado.

Perceba que Filme deve ter quatro informações; Id, Titulo, descrição e ano de lançamento do filme. Já Categoria deverá ter: um identificador e o nome da categoria.

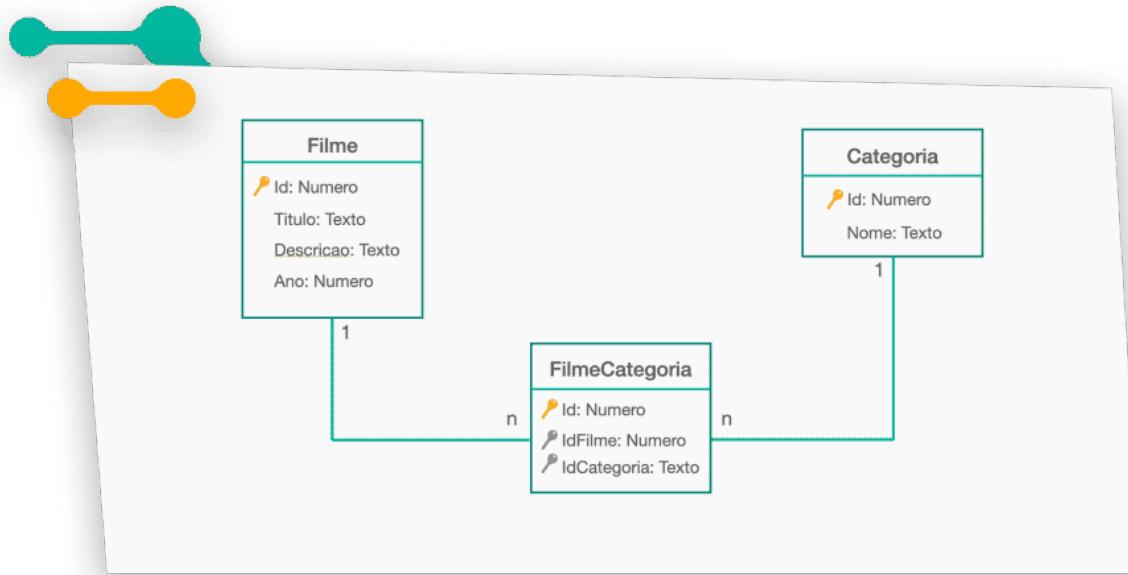


Essa visão nos prepara para que mais a frente seja possível criar efetivamente as entidades de banco de dados para soluções Low-Code.

Modelo Lógico

Esse modelo procura trazer mais detalhes sobre os conceitos dos dados, assim como as relações complexas identificadas anteriormente no modelo conceitual.

Uma forma usada por esse modelo para conceituar melhor os dados, é informar o tipo de cada um, como por exemplo, se é texto ou número. Perceba que na imagem ao lado há um exemplo que mostra um refino no relacionamento entre Filme e Categoria, considerados no modelo conceitual.



Para que seja possível representar em entidades o **relacionamento n para n**, que no nosso contexto é representado por cada Filme poder estar relacionado com várias Categorias, assim como cada categoria poder estar relacionada com vários Filmes, há a necessidade de criar uma entidade que esteja entre elas. Assim, é possível mapear toda relação de filme com categoria.

Por isso, que no modelo lógico apresentado, há uma nova entidade chamada **FilmeCategoria**. Perceba que ela possui **três identificadores**:

1

Id

Indica o identificador único de cada registro da entidade. Esse tipo de

identificador único por registro é também conhecido como ***Primary key*** no contexto de banco de dados.

2

idFilme

Faz referencia a um identificador filme existente.

3

idCategoria

Referencia um identificador de uma categoria existente.

Tanto idFilme, como idCategoria, fazem referência a identificadores de registros de outras entidades, sendo também conhecidos como ***Foreign Keys*** no contexto de banco de dados.

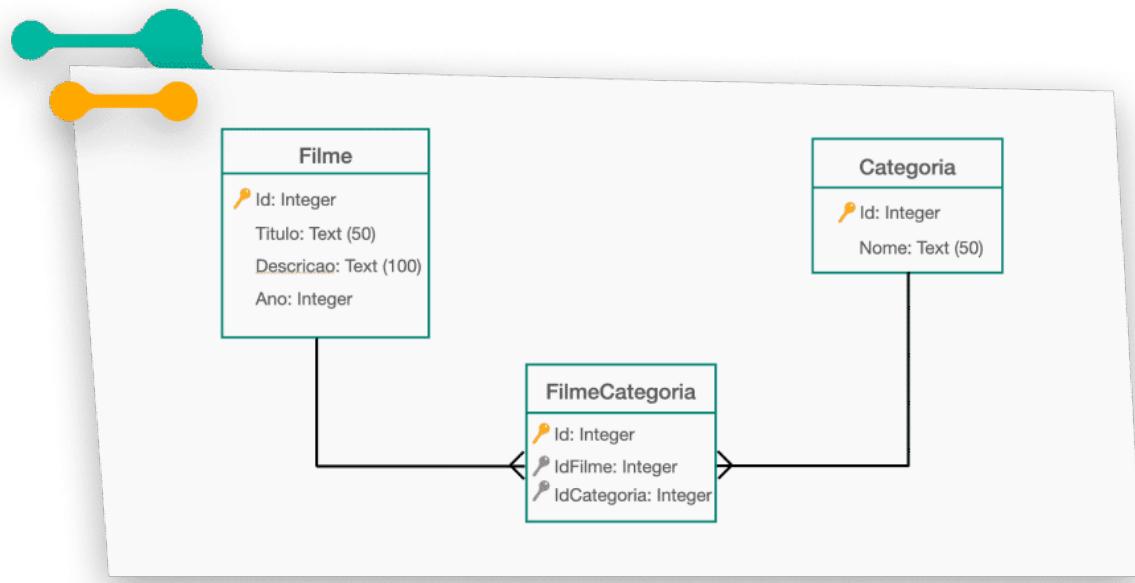
Perceba que esse modelo orienta ainda mais como é possível criar entidades de banco de dados na plataforma da OutSystems.

Modelo Físico

Esse é o modelo mais fiel e detalhado que será seguido e representado no desenvolvimento.

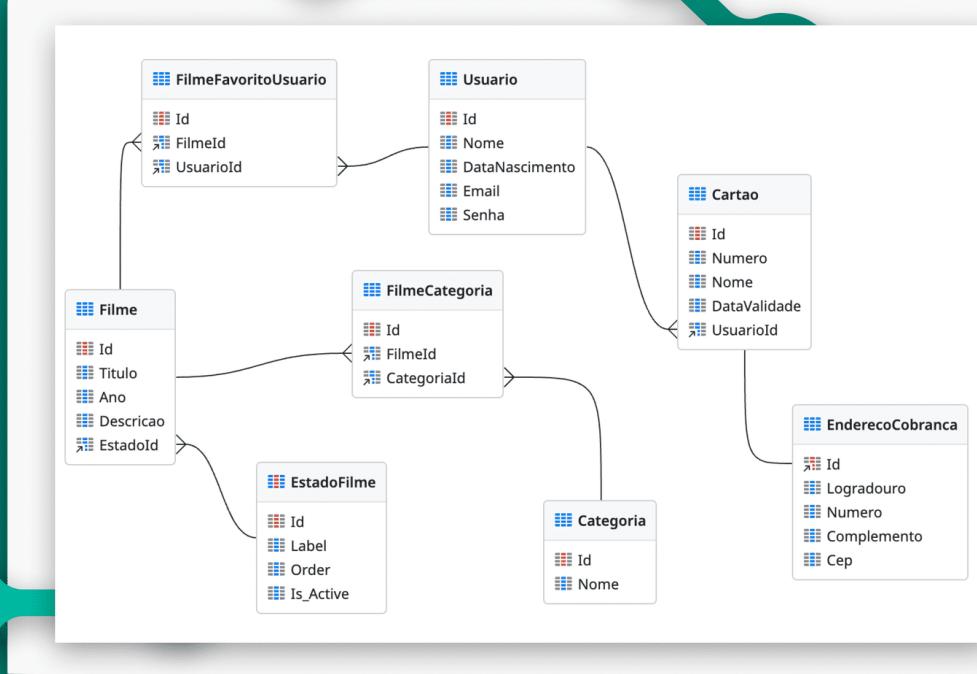
No mercado há diversas ferramentas que oferecem suporte para criar os diferentes tipos de modelos apresentados, e que dependendo da ferramenta podem haver variações visuais.

No modelo físico que está sendo apresentado abaixo, perceba que no lugar de usar as cardinalidades, ele usa uma abordagem diferente para indicar o lado do relacionamento n. Além disso, o modelo também informa o tipo de dado que realmente será considerado em cada atributo da entidade.



Tipos de Relacionamentos entre Entidades de Banco de Dados

**Tipos de Relacionamentos
entre Entidades de Banco
de Dados**



O Service Studio oferece um recurso visual que permite criar, visualizar e editar relacionamentos entre entidades de banco de dados.

Na imagem ao lado, perceba que há 7 entidades criadas, assim como seu atributos e relacionamentos.

Para permitir que esses relacionamentos existam, é necessário que cada entidade tenha um identificador único definido.

A plataforma Low-Code da OutSystems permite a representação de três tipos de relacionamentos:

- 1 1 para muitos;
- 2 1 para 1;
- 3 Muito para muitos.

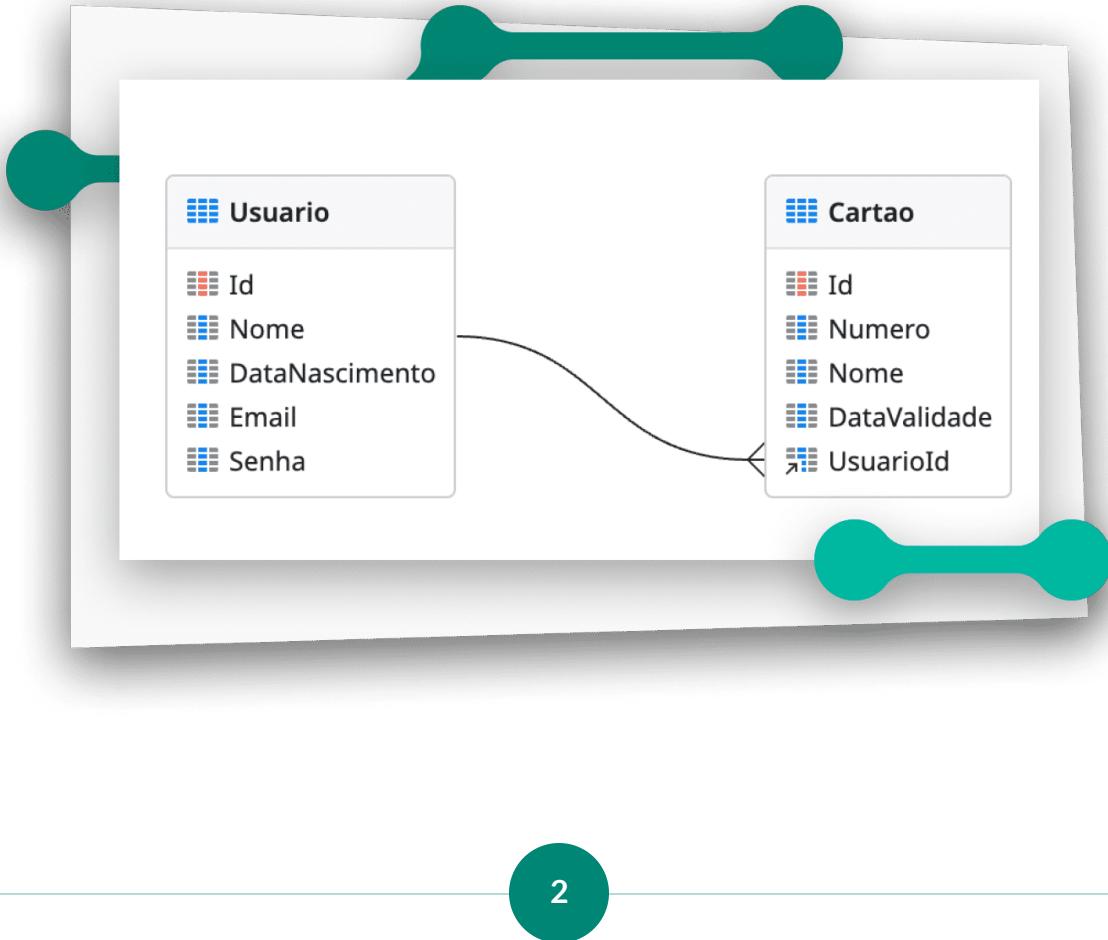


Relacionamento 1 para Muitos

Esse é o tipo de relacionamento que um registro da entidade A está associada a muitos registros da entidade B, enquanto que um registro de B está relacionado a somente um registro de A. Para exemplificar essa ideia, abaixo, há uma imagem que representa esse tipo de relacionamento, considerando as entidades Usuário e Cartão.

A entidade Usuário é onde estarão salvas as informações dos usuários que poderão acessar algum sistema desenvolvido. Enquanto que a entidade Cartão indica qual cartão de crédito estará registrado por usuário.

Nesse exemplo está sendo informado que cada usuário poderá ter diferentes cartões de crédito vinculados a ele. Além disso, cada cartão registrado deverá estar vinculado a somente um usuário. Com isso temos o relacionamento 1 para muitos.



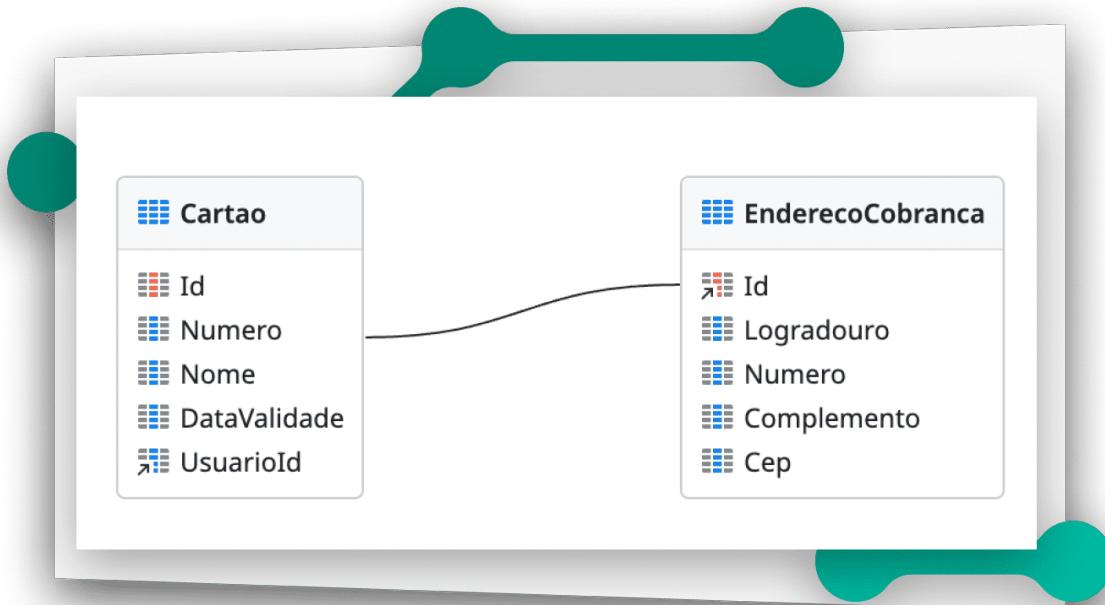
2

Relacionamento 1 para 1

Nesse tipo de relacionamento, um registro da entidade A está associada a somente um registro da entidade B, assim como um registro de B está associada a somente um registro de A. Para exemplificar essa ideia, na imagem abaixo há um exemplo com as entidades Cartão e Endereço de Cobrança.

A entidade Cartão segue representando os cartões de crédito registrados, enquanto que a outra entidade representa o endereço de cobrança de algum cartão.

Nesse relacionamento está sendo informado que para cada cartão há 1 endereço de cobrança relacionado. Além disso, para cada endereço de cobrança, há somente 1 cartão vinculado a ele. Dessa forma o relacionamento caracterizasse como 1 para 1.



Quando for criado um relacionamento 1 para 1 na OutSystems, o desenvolvedor deverá encontrar as seguintes opções em uma propriedade chamada Delete Rule: **Protect**, **Delete** e **Ignore**. Ela aparece quando é selecionado o identificador da entidade secundária do relacionamento, que no exemplo anterior é a entidade **EnderecoCobranca**.

Abaixo há a explicação de cada opção oferecida:

● **Protect**

Indica que é impedida a exclusão do registro principal enquanto houver algum outro registro relacionado a ele. No nosso exemplo acima a entidade principal é Cartão.

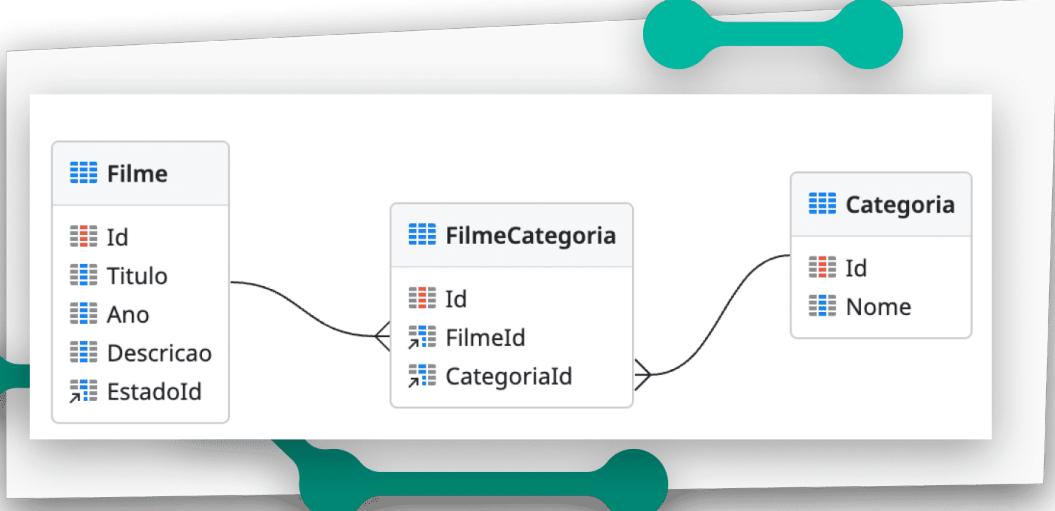
- **Delete**
Informa que quando o registro da entidade principal for deletado, todos os registros relacionados a ele também serão apagados.
- **Ignore**
Permite deletar os registros da entidade principal mantendo os registros relacionados.

3

Relacionamento Muito para Muitos

Nesse tipo de relacionamento, um registro da entidade A pode estar associado a muitos registros da entidade B, assim como um registro de B pode estar associado a muitos registros de A. Para exemplificar essa ideia, na imagem abaixo há um exemplo informando que cada Filme pode estar relacionado com diversas categorias de filme. Já cada categoria pode também estar relacionada com diversos filmes.

Para representar esse tipo de relacionamento muito para muitos, há a necessidade de criar uma entidade intermediária, assim como ilustrado. Perceba que há uma entidade chamada **FilmeCategoria**, que representa todas as possíveis combinações de Filme e Categoria. Por isso que nessa entidade há **FilmId** e **CategorId** para permitir tal representação.



Conhecendo Índices

Conhecendo Índices

Índices são estruturas de dados voltadas a acelerar as consultas em algum banco de dados.

Para melhor entendermos essa definição, imagine um livro com diversas informações, dividido, por exemplo, por capítulos e seções. Caso esteja procurando algo no livro, o que provavelmente seria feito para ajudar nessa busca?

Geralmente o índice ou sumário do livro são usados, pois oferecem uma visão da página que inicia algum capítulo, assim como alguma seção.

Com isso, considere que cada entidade de banco de dados da plataforma OutSystems pode ser considerada um livro com diversas informações. Caso não tenha um índice disponível, e esteja, por exemplo, procurando a partir do Número de uma Identidade, uma pessoa registrada, como seria para encontrá-la?

Provavelmente teríamos que olhar informação por informação até encontrar a Identidade desejada. É como se fosse folhear página por página de um livro passando por todo conteúdo até encontrar o que deseja.



Imagine o trabalho!

Caso nessa entidade hajam mais de 10.000 registros presentes, imagine quantas informações teriam que ser analisadas para conseguir encontrar o registro desejado. Para facilitar essa busca, imagine se os números das identidades estivessem ordenados em um índice, do número menor para o maior.

Encontrar a identidade registrada seria mais rápida!



Voltando a analogia do livro, tendo um índice com essa ideia, ao encontrar o número da identidade no índice, você poderia ir direto para a página correspondente e ler as informações da pessoa desejada.

Essa mesma ideia é aplicada ao índice de banco de dados.

Três Vantagens dos Índices

Abaixo, encontram-se 3 vantagens relacionadas ao uso de índices.

- Aumentar a velocidade da execução de consultas para recuperar registros desejados.
- Índices são úteis na situação em que percebesse a necessidade de utilizar por muitas vezes dados seguindo algum critério de ordenação. Caso isso seja percebido, índices evitam o custo de ordenação.
- Contribuir para a implementação de restrições, como por exemplo, garantir que algum valor será realmente único para algum atributo de

uma entidade.

Três Desvantagens dos Índices

Abaixo, encontram-se 3 desvantagens relacionadas ao uso de índices.

- **Piorar o desempenho na escrita de dados nas entidades.** Toda vez que algum atributo chave for alterado e que é usado em algum índice, tal índice terá mudanças.
- **Aumento de espaço, como memória e disco, para armazenamento do banco de dados.** Caso alguma solução tenha muitos índices, o espaço pode talvez ser maior do que a entidade original usada por ele.
- **Maior esforço para manter os índices.** Conforme mudanças possam acontecer com o passar do tempo no banco de dados, poderá haver a necessidade de aplicar mudanças nos índices para atender a evolução do software correspondente.

Perceba que índice é um recurso muito útil, mas não deve ser aplicado de forma excessiva. Procure analisar situações que realmente valem a pena criar algum índice e assim trazer benefícios para o seu software.

Encerramento

Clique na seta abaixo para fazer o download deste E-book.



File Attachment Block

No file added





PARABÉNS

Você completou este Módulo