3 Mapeamento de objetos para o Modelo Relacional

Um ladrão rouba um tesouro, mas não furta a inteligência.
Uma crise destrói um herança, mas não uma profissão.
Não importa se você não tem dinheiro, você é uma pessoa rica,
pois possui o maior de todos os capitais: a sua inteligência.
Invista nela. ***Estude****!.*
Augusto Cury

A tecnologia de orientação a objetos é consolidada como a forma mais usual para o desenvolvimento de softwares. Já, quando pensamos em banco de dados, os bancos relacionais forma os que tiveram maior exito e, sem dúvida, os Sistemas Gerenciadores de Banco de dados Relacionais (SGBDR) dominam o mercado.

No entanto, essas duas tecnologias surgiram por meio de princípios teóricos muito diferentes. Segundo Bezerra 2016 o descasamento de informações *(impedance mismatch)* é um termo utilizado para denotar o problema das diferenças entre as representações do modelo de objetos e do modelo relacional.

"Os princípios básicos do paradigma da orientação a objetos e do modelo relacional são bastante diferentes. No modelo de objetos, os elementos (objetos) correspondem a abstrações de comportamento. No modelo relacional, os elementos correspondem a dados no formato tabular [Bezerra 2016]". Assim, neste capítulo, vamos apresentamos alguns detalhes a respeito dos diversos aspectos relativos ao mapeamento de objetos para um mecanismo de armazenamento persistente.

3.1 Projeto de banco de dados

Uma das principais atividades do projeto é o desenvolvimento do banco de dados. Essa atividade, durante o processo, normalmente é chamada de projeto de banco de dados. Tal projeto envolve diversas atividades, tais como:

- Construção do esquema do banco de dados.
- Criação de índices2 para agilizar o acesso aos dados armazenados.
- Definição das estruturas de dados a serem utilizadas no armazenamento físico dos dados.

- Definição de visões sobre as dados armazenados.
- Atribuição de direitos de acesso (que usuários podem acessar que recursos).
- Definição de políticas de backup dos dados.

3.1.1 Conceitos básicos do modelo de dados relacionais

O modelo relacional, segundo Bezerra 2016, se fundamenta no conceito de relação. De forma bastante simplista, pode-se pensar em uma relação como uma tabela, composta de linhas e de colunas. Cada relação possui um nome. Cada coluna de uma relação possui um nome e um domínio.

Departamento

id	sigla	nome	idGerente
13	RH	Recursos Humanos	5
14	INF	Informática	2
15	RF	Recursos Financeiros	6

Figura 23 – Exemplo de relações em um banco de dados

Fonte: [Bezerra 2016]

3.1.1.1 Chave Primária

Um conceito importante no modelo relacional é o de **chave primária**. Uma chave primária é uma coluna ou conjunto de colunas cujos valores podem ser utilizados para identificar unicamente cada linha de uma relação [Bezerra 2016]. Note que a Figura 23 possue uma coluna chamada id. Essa coluna é a chave primária ds relação na qual aparece.

3.1.1.2 Chave Estrangéira

Outro conceito importante do modelo relacional é o de **chave estrangeira**. Linhas de uma relação podem estar associadas às de outras relações. Estas associações são representadas pela única maneira possível, considerando-se os recursos de notação do modelo relacional: deve existir, em uma das duas relações, uma coluna cujos valores fazem referência aos de uma coluna da outra relação. Na terminologia do modelo relacional, esta coluna de referência é denominada chave estrangeira. Em nossa Figura 23 a chave estrangeira é representada pelo campo idGerente, o qual apresenta-se tracejado.

3.2 Mapeamento de objetos para o modelo relacional

Quando utilizamos um SGBD para gerenciar nossa base de dados e nosso armazenamento persistente de informações para um sistema de software orientado a objetos, há a necessidade de se realizar o mapeamento dos valores de atributos de objetos persistentes do sistema para tabelas.

Assim, em nossa próxima sessão vamos aprender o procedimento de mapeamento de diversos elementos do modelo de classes para o modelo relacional.

3.2.1 Classe e seus atributos

Segundo Bezerra 2016, classes são mapeadas para relações. O caso mais simples é o de mapear cada classe como uma relação e cada um de seus atributos como uma coluna da relação correspondente. No entanto, muito frequentemente não há uma correspondência unívoca entre classes e relações. Pode ser que várias classes sejam mapeadas para uma única relação ou que uma classe seja mapeada para várias relações.

Para atributos o que vale de forma geral é que um atributo será mapeado para uma ou mais colunas. Lembre-se, também, de que nem todos os atributos são persistentes. Por exemplo, pode ser que uma classe Pedido tenha um atributo derivado, total, utilizado para guardar o valor total a ser pago por um pedido, mas que este atributo não seja armazenado no banco de dados.

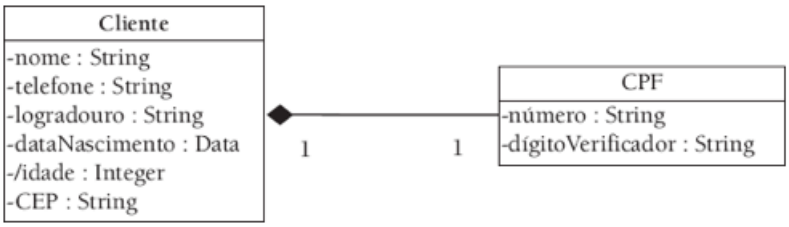


Figura 24 – Classe cliente é CPF

Fonte: [Bezerra 2016]

Considere a classe apresentada na Figura 24. A Figura 25 exibe duas alternativas de mapeamento possíveis para essa classe. Note que, em ambos os casos, o atributo derivado idade não é mapeado. Além disso, nas duas alternativas, as classes **Cliente** e **CPF** são mapeadas para uma única relação. No entanto, na primeira alternativa, o CEP de um cliente é mapeado para uma relação separada e o atributo CEP da classe é dividido em duas partes (número e dígitoVerificador).

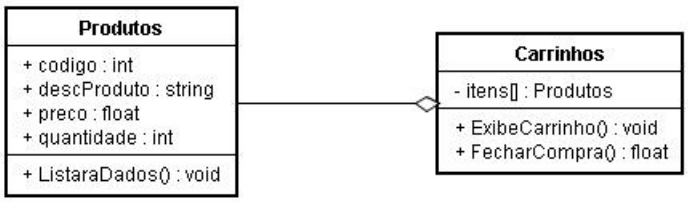
1ª	Cliente(id, CPF, nome, telefone, logradouro, dataNascimento, idCEP) CEP(id, número, sufixo)
2ª	Cliente(id, nome, telefone, logradouro, dataNascimento, CPF, CEP)

Figura 25 – Classe cliente é CPF

Fonte: [Bezerra 2016]

3.3 Exercício de fixação

Utilizando os conceitos até o momento apresentados, realize o mapeamento do digrama abaixo.



3.3.1 Associações

Para que possamos mapear associações, nós utilizarmos o conceito de **chave estrangeira**. Uma chave estrangeira, como descrito na sub-sessão 3.1.1.2, são utilizadas para relacionar linhas de relações diferentes ou linhas de uma mesma relação. Para isso, há três casos para mapeamento de associações, cada um correspondente a um tipo de conectividade.

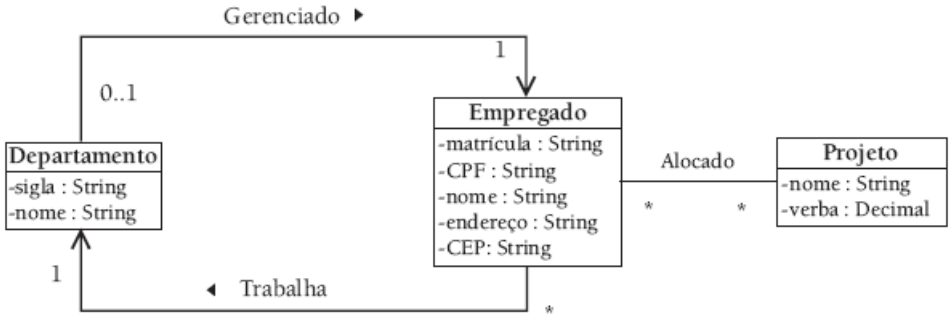


Figura 26 – Diagrama de classes ilustrando os três tipos de conectividade em associações

Fonte: [Bezerra 2016]

3.3.1.1 Associação um-para-um

Quando há uma associação um-para-um entre **Ca** e **Cb**, deve-se adicionar uma chave estrangeira em uma das duas relações para referenciar a chave primária da outra relação.

Departamento(id, sigla, nome, idEmpregadoGerente)

Empregado(id, matrícula, CPF, nome, endereço, CEP)

Contudo, há casos que a associação um-para-um pode ser mapeada para uma única relação, como no caso do diagrama apresentado na Figura 24

3.3.1.2 Associação um-para-muitos

Em uma associação um-para-muitos entre objetos de **Ca** e de **Cb**, seja **Ca** a classe na qual cada objeto se associa com muitos objetos da classe **Cb**. Neste caso, deve-se adicionar uma chave estrangeira em **Ca** para referenciar a chave primária de **Cb** [Bezerra 2016].

Como exemplo, considere novamente a Figura 26, na qual se apresenta a associação um para muitos Trabalha. A seguir é apresentada uma extensão do mapeamento anterior considerando essa associação.

Departamento(id, sigla, nome, idEmpregadoGerente)

Empregado(id, matrícula, CPF, nome, endereço, CEP, idDepartamento)

3.3.1.3 Associações de conectividade muitos para muitos

Quando há uma associação de conectividade muitos para muitos entre objetos de Ca e de Cb, uma relação de associação deve ser criada. Uma relação de associação tem o objetivo de representar a associação muitos para muitos entre duas ou mais relações [Bezerra 2016].

Departamento(id, sigla, nome, idEmpregadoGerente)

Empregado(id, matrícula, CPF, nome, endereço, CEP, idDepartamento)

Alocação(id, idProjeto, idEmpregado, nome, verba)

Projeto(id, nome, verba)

Assim, de forma geral, independente da conectividade, o conceito de chave estrangeira é sempre utilizado quando realizamos o mapeamento. Esta técnica é a mesma utilizada para a **agregação** ou para a **composição**. Contudo, para que se possa alcançar o comportamento esperado, devemos utilizar técnicas que envolvem os gatilhos (**triggers**) e procedimentos armazenados (**stored procedures**), os quais não serão tratados neste capítulo.

3.3.1.4 Associação reflexiva

Uma associação reflexiva, ou também conhecida como auto-associação, é um tipo especial de associação. Contudo, o mapeamento aplicado a associações é igualmente ao aplicado em outras associações.

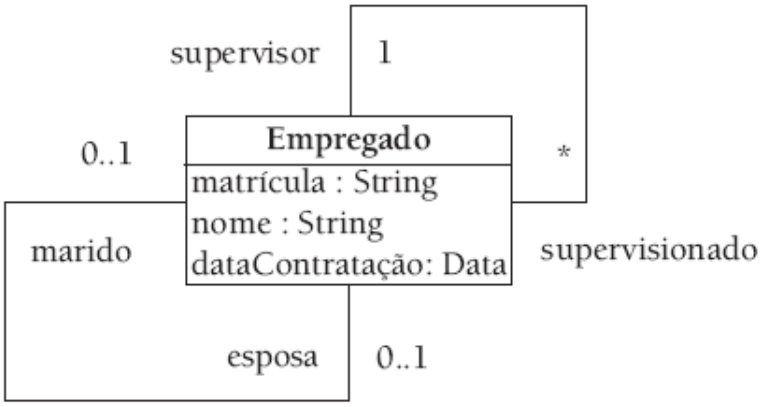


Figura 27 – Associações reflexivas entre objetos da classe Empregado

Fonte: [Bezerra 2016]

Um possível mapeamento desse diagrama é apresentado a seguir. Note que as chaves estrangeiras idCônjunge e idSupervisor foram definidas na relação Empregado como era esperado, visto que ambas as associações são reflexivas.

Empregado(id, matrícula, nome, dataContratação, idCônjunge, idSupervisor)

3.3.1.5 Associação ternária

Associações ternárias, ou também conhecidas como associações n-árias, podem ser mapeadas por meio de um procedimento semelhante ao utilizado para associações binárias de conectividade muitos para muitos. Uma relação para representar a associação é criada e são adicionadas nelas chaves estrangeiras para as **n** classes participantes da associação. Se a associação ternária possuir uma classe associativa, os atributos dela são mapeados como colunas da relação de associação [Bezerra 2016].

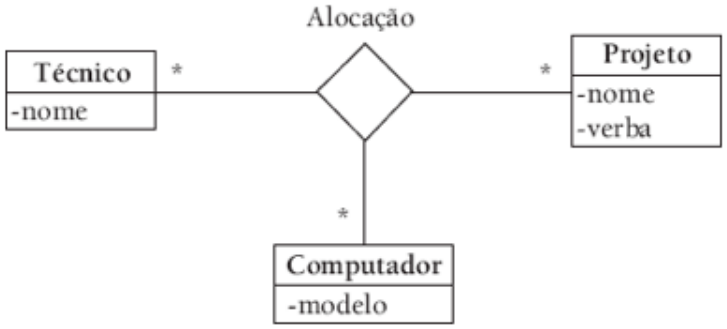


Figura 28 – Associação terminária/N-ária

Fonte: [Bezerra 2016]

Generalização

Há basicamente três alternativas de se mapear relacionamentos de generalização.

1. Uma relação para cada classe da hierarquia;
2. Uma relação para toda a hierarquia;
3. Uma relação para cada classe concreta da hierarquia.

Assim, vamos aplicar as três formas de mapeamento para o diagrama de classe abaixo.

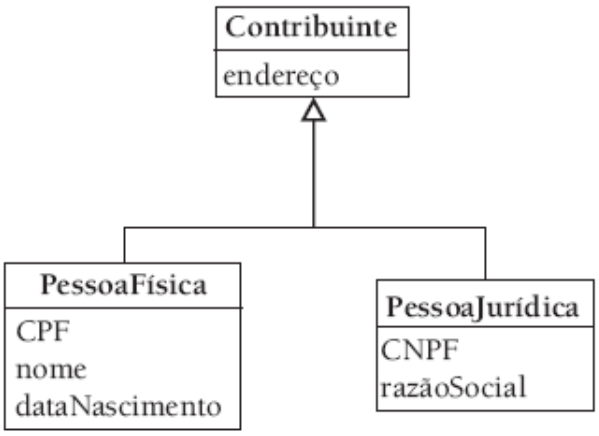


Figura 29 – Hierarquia de generalização

Fonte: [Bezerra 2016]

Realizando o mapeamento por meio das três alternativas possíveis, obtemos o seguinte resultado:

Alternativa 1

Contribuinte(id, endereço)

PessoaFísica(id, nome, dataNascimento, CPF, idContribuinte)

PessoaJurídica(id, CNPJ, razãoSocial, idContribuinte)

Alternativa 2

Pessoa(id, nome, endereço, dataNascimento, CPF, CNPJ, razãoSocial, tipo)

Alternativa 3

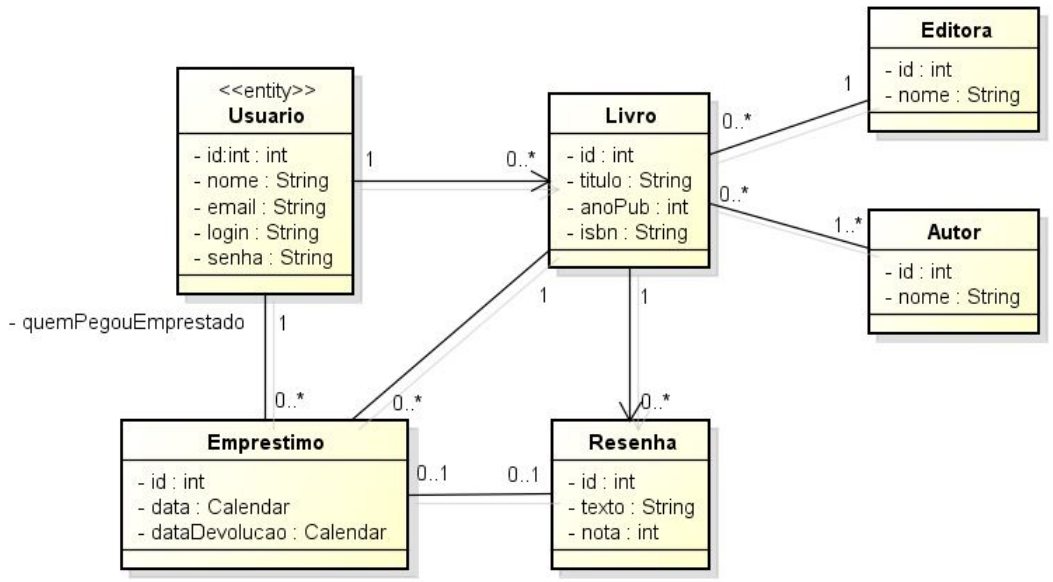
PessoaFísica(id, dataNascimento, nome, endereço, CPF)

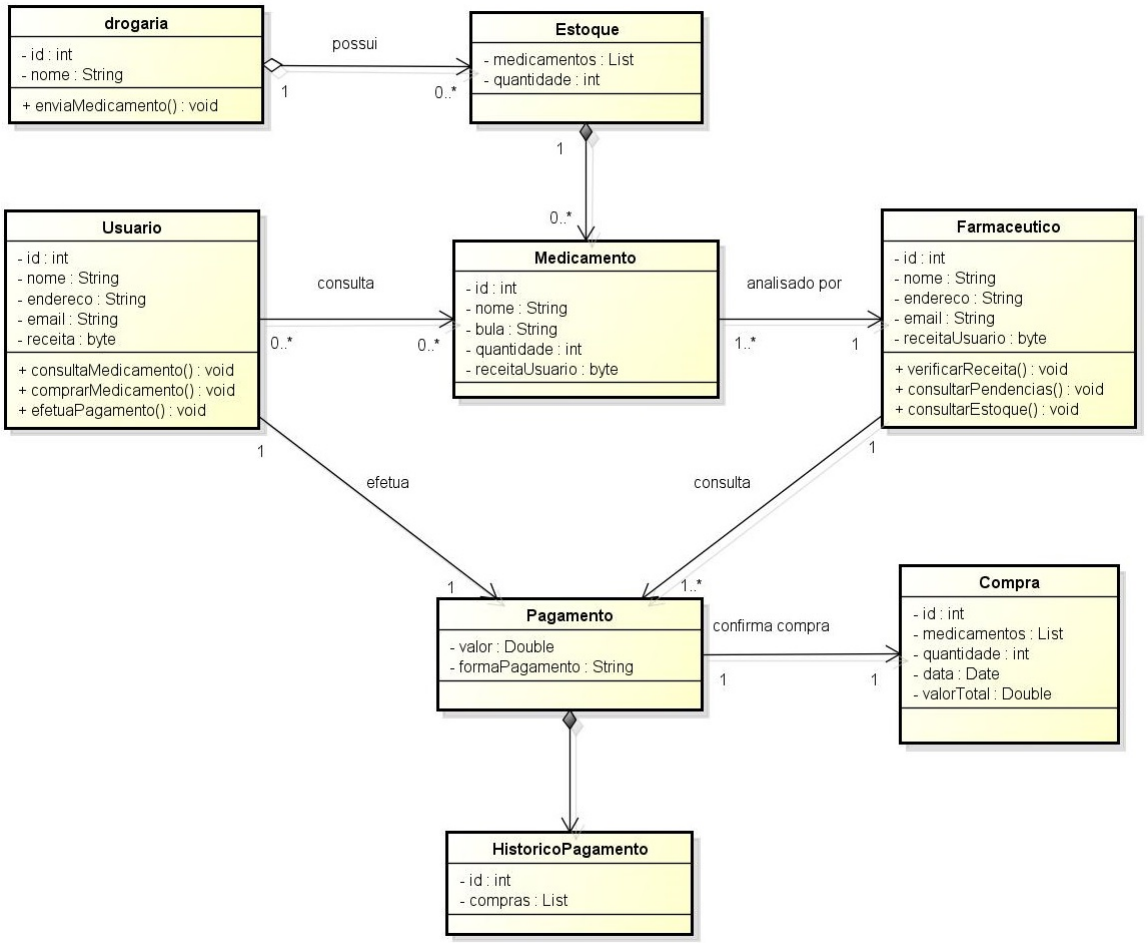
PessoaJurídica(id, CNPJ, endereço, razãoSocial)

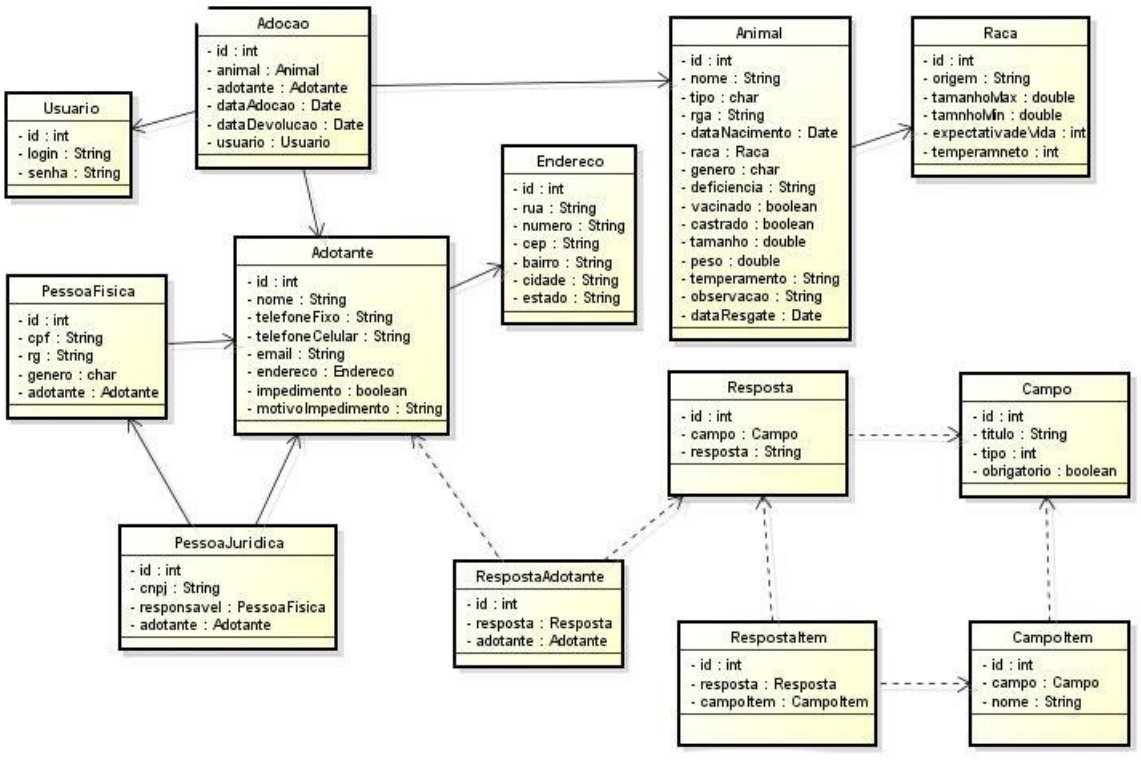
3.4 Exercício de fixação

Utilizando os conceitos até o momento apresentados, realize o mapeamento dos diagramas abaixo.

1.




drogaria
- id : int
- nome : String
+ enviaMedicamento() : void
possui
1
0..*
Estoque
- medicamentos : List
- quantidade : int
1
0..*
Usuario
- id : int
- nome : String
- endereco : String
- email : String
- receita : byte
+ consultaMedicamento() : void
+ comprarMedicamento() : void
+ efetuaPagamento() : void
consulta
0..*
0..*
Medicamento
- id : int
- nome : String
- bula : String
- quantidade : int
- receitaUsuario : byte
analisado por
1..*
1
Farmaceutico
- id : int
- nome : String
- endereco : String
- email : String
- receitaUsuario : byte
+ verificarReceita() : void
+ consultarPendencias() : void
+ consultarEstoque() : void
1
efetua
consulta
1
1..*
Pagamento
- valor : Double
- formaPagamento : String
confirma compra
1
1
Compra
- id : int
- medicamentos : List
- quantidade : int
- data : Date
- valorTotal : Double
HistoricoPagamento
- id : int
- compras : List


Adocao
- id : int
- animal : Animal
- adotante : Adotante
- dataAdocao : Date
- dataDevolucao : Date
- usuario : Usuario
Usuario
- id : int
- login : String
- senha : String
Animal
- id : int
- nome : String
- tipo : char
- rga : String
- dataNacimento : Date
- raca : Raca
- genero : char
- deficiencia : String
- vacinado : boolean
- castrado : boolean
- tamanho : double
- peso : double
- temperamento : String
- observacao : String
- dataResgate : Date
Raca
- id : int
- origem : String
- tamanholvlax : double
- tamnholvlin : double
- expectativadeVida : int
- temperamneto : int
Endereco
- id : int
- rua : String
- numero : String
- cep : String
- bairro : String
- cidade : String
- estado : String
Adotante
- id : int
- nome : String
- telefoneFixo : String
- telefoneCelular : String
- email : String
- endereco : Endereco
- impedimento : boolean
- motivoImpedimento : String
PessoaFisica
- id : int
- cpf : String
- rg : String
- genero : char
- adotante : Adotante
Resposta
- id : int
- campo : Campo
- resposta : String
Campo
- id : int
- titulo : String
- tipo : int
- obrigatorio : boolean
PessoaJuridica
- id : int
- cnpj : String
- responsavel : PessoaFisica
- adotante : Adotante
RespostaAdotante
- id : int
- resposta : Resposta
- adotante : Adotante
RespostaItem
- id : int
- resposta : Resposta
- campoItem : CampoItem
CampoItem
- id : int
- campo : Campo
- nome : String

Referências

BEZERRA, E. *Princípios de Análise e Projeto de Sistema com UML*. [S.l.]: Elsevier Brasil, 2016. v. 3. Citado 10 vezes nas páginas 13, 14, 15, 23, 24, 25, 26, 27, 28 e 29.

GUEDES, G. T. *UML 2-Uma abordagem prática*. [S.l.]: Novatec Editora, 2018. Citado 2 vezes nas páginas 13 e 15.

PRESSMAN, R.; MAXIM, B. *Engenharia de Software-8ª Edição*. [S.l.]: McGraw Hill Brasil, 2016. Citado na página 4.

RUIZ, E. E. S. *IBm1030 Programação Orientada a Objetos*. 2008. Disponível em: <http://dcm.ffclrp.usp.br/~evandro/ibm1030/constru/heranca.html>. Citado na página 6.

SILVA, D. Lucas da et al. Ontologias e unified modeling language: uma abordagem para representação de domínios de conhecimento. v. 10, 10 2009. Citado na página 8.

SOMMERVILLE, I. *Engenharia de Software. 6ª*. [S.l.: s.n.], 2003. Citado 3 vezes nas páginas 2, 3 e 4.