Sistema digital PetFera

Gerado por Doxygen 1.8.13

Sumário

Capítulo 1

Lista de Futuras Atividades

Classe Petshop

Cadastro de Especies

- Lisenças para animais (Silvestre... Nativo)
- Classificações de status de extinção para animais com base na espécie

Capítulo 2

Índice Hierárquico

2.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Capítulo 3

Índice dos Componentes

3.1 Lista de Componentes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Capítulo 4

Classes

4.1 Referência da Classe Anfibio

Classificação base para Anfibios.

```
#include <anfibio.hpp>
```

Diagrama de Hierarquia para Anfibio:

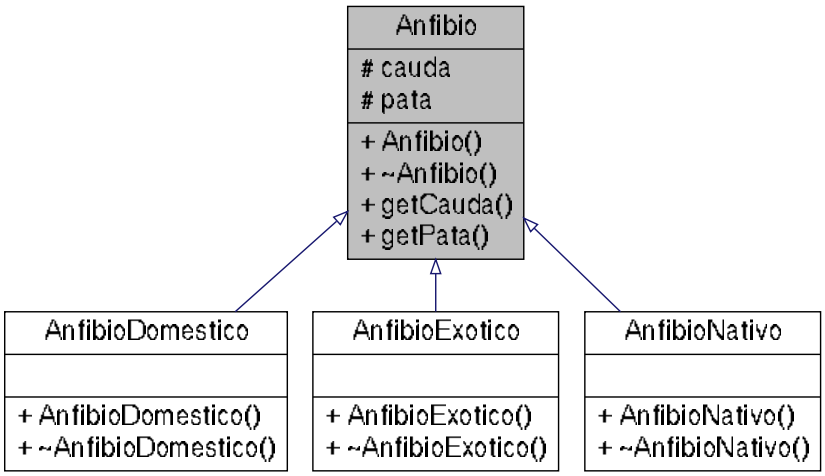
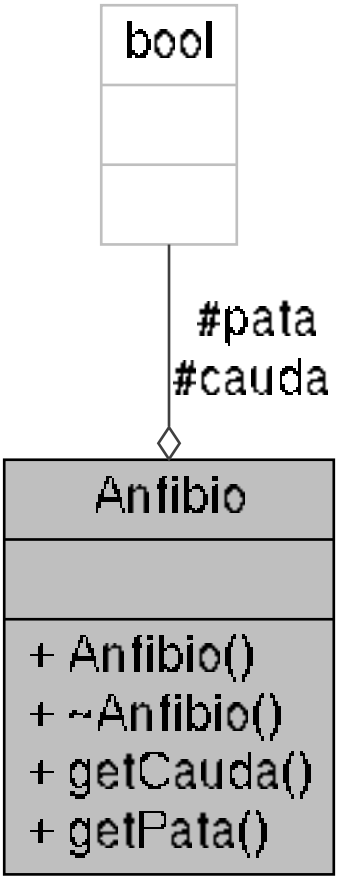


Diagrama de colaboração para Anfibio:



Métodos Públicos

- **Anfibio** (bool cauda, bool pata)
- bool **getCauda** () const
- bool **getPata** () const

Atributos Protegidos

- bool **cauda**
- bool **pata**

4.1.1 Descrição Detalhada

Classificação base para Anfibios.

A classe serve como base para os animais que se enquadram na Classe. Tendo herdeiros com base na Categoria:

- Domestico
- Nativo
- Exotico

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/anfibio/anfibio.hpp

4.2 Referência da Classe AnfibioDomestico

Implementação de animal com Classe e Categoria.

`#include <anfibio_domestico.hpp>`

Diagrama de Hierarquia para AnfibioDomestico:

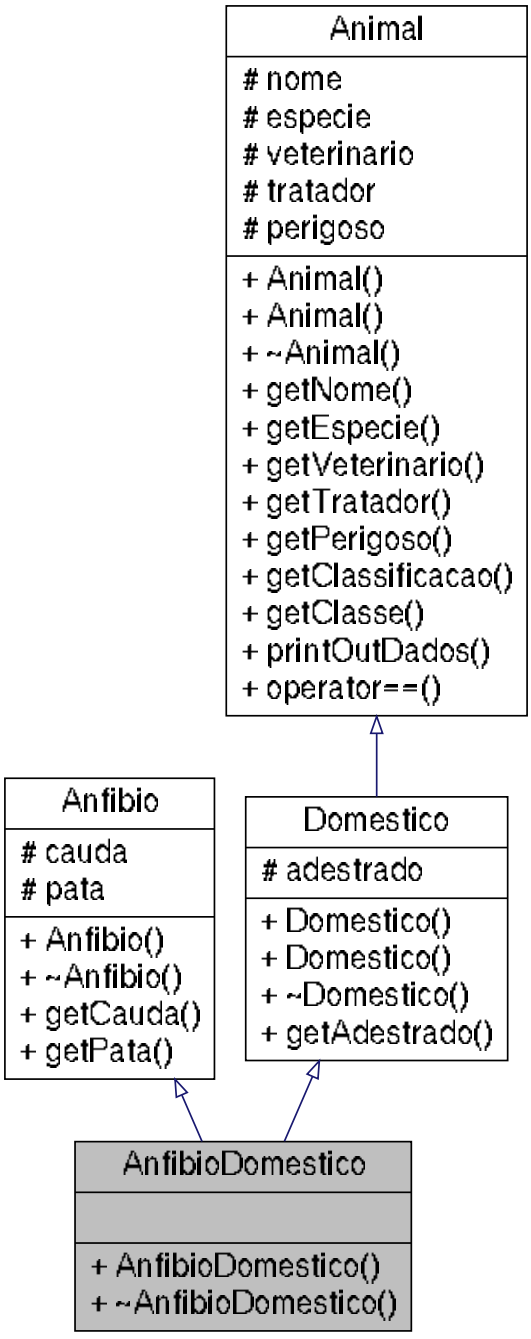
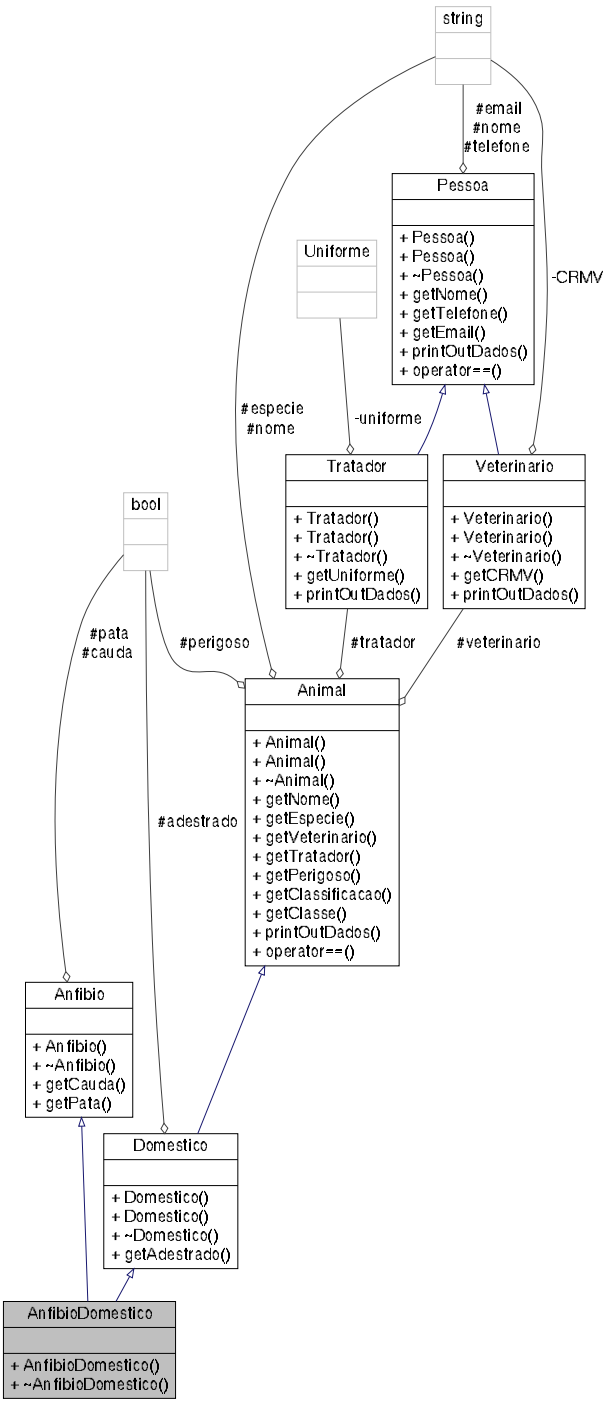


Diagrama de colaboração para AnfibioDomestico:



Métodos Públicos

- **AnfibioDomestico** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, bool adestrado, bool cauda, bool pata)

Outros membros herdados

4.2.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Anfibio do tipo Domestico.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/anfibio/anfibio_domestico.hpp

4.3 Referência da Classe AnfibioExotico

Implementação de animal com Classe e Categoria.

```
#include <anfibio_exotico.hpp>
```

Diagrama de Hierarquia para AnfibioExotico:

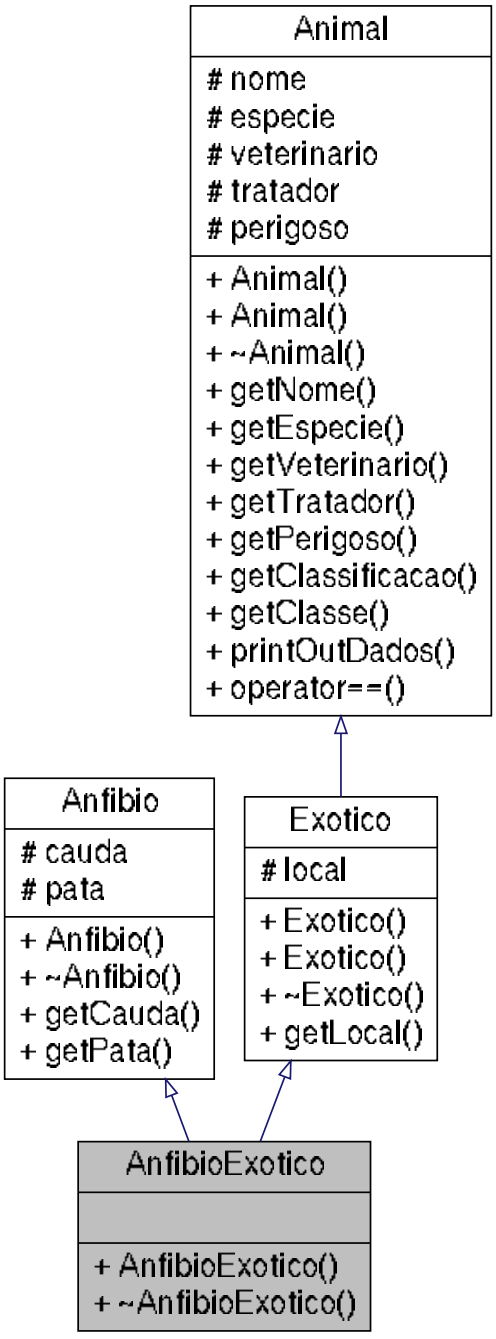
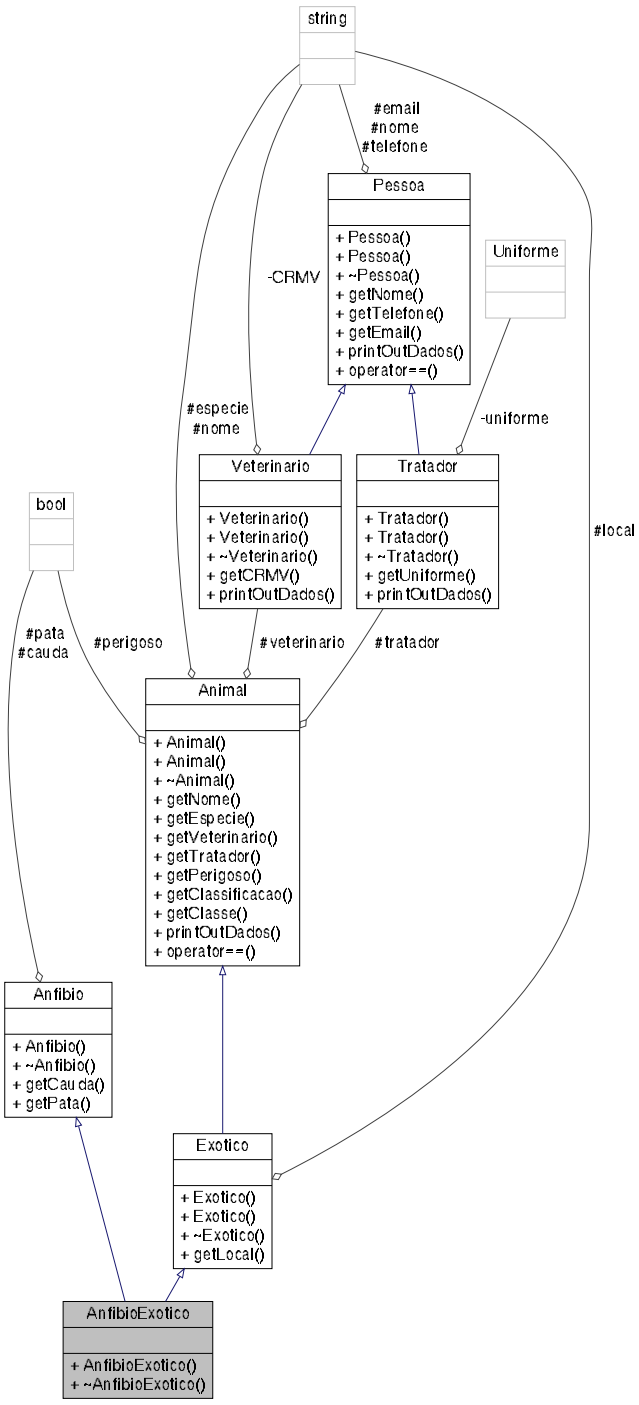


Diagrama de colaboração para AnfibioExotico:



Métodos Públicos

- **AnfibioExotico** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string local, bool cauda, bool pata)

Outros membros herdados

4.3.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Anfibio do tipo Exotico.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/anfibio/anfibio_exotico.hpp

4.4 Referência da Classe AnfibioNativo

Implementação de animal com Classe e Categoria.

```
#include <anfibio_nativo.hpp>
```

Diagrama de Hierarquia para AnfibioNativo:

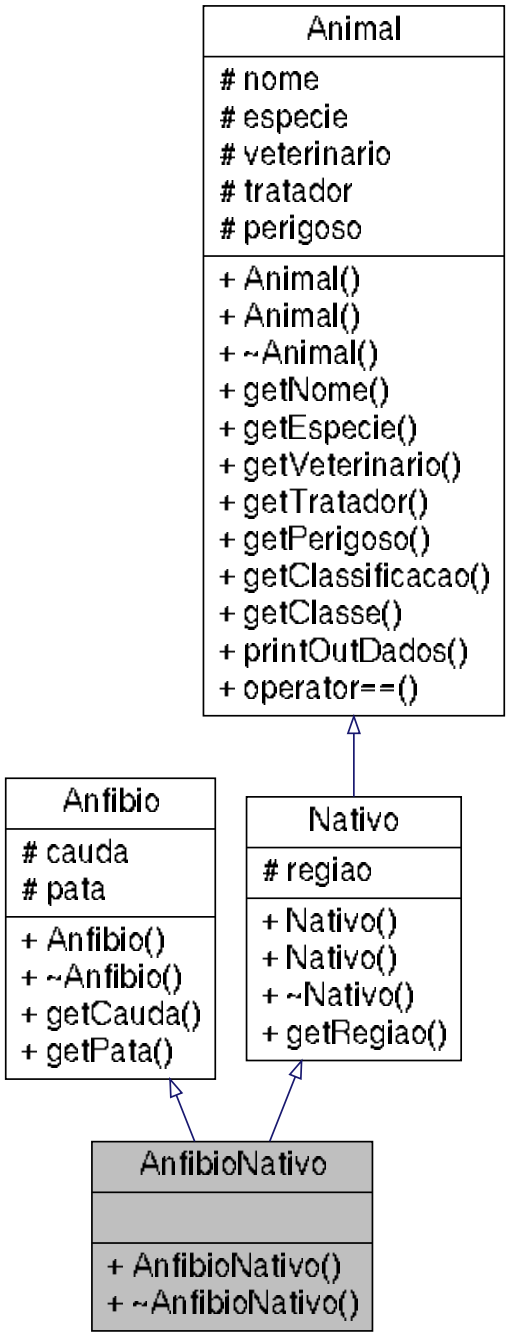
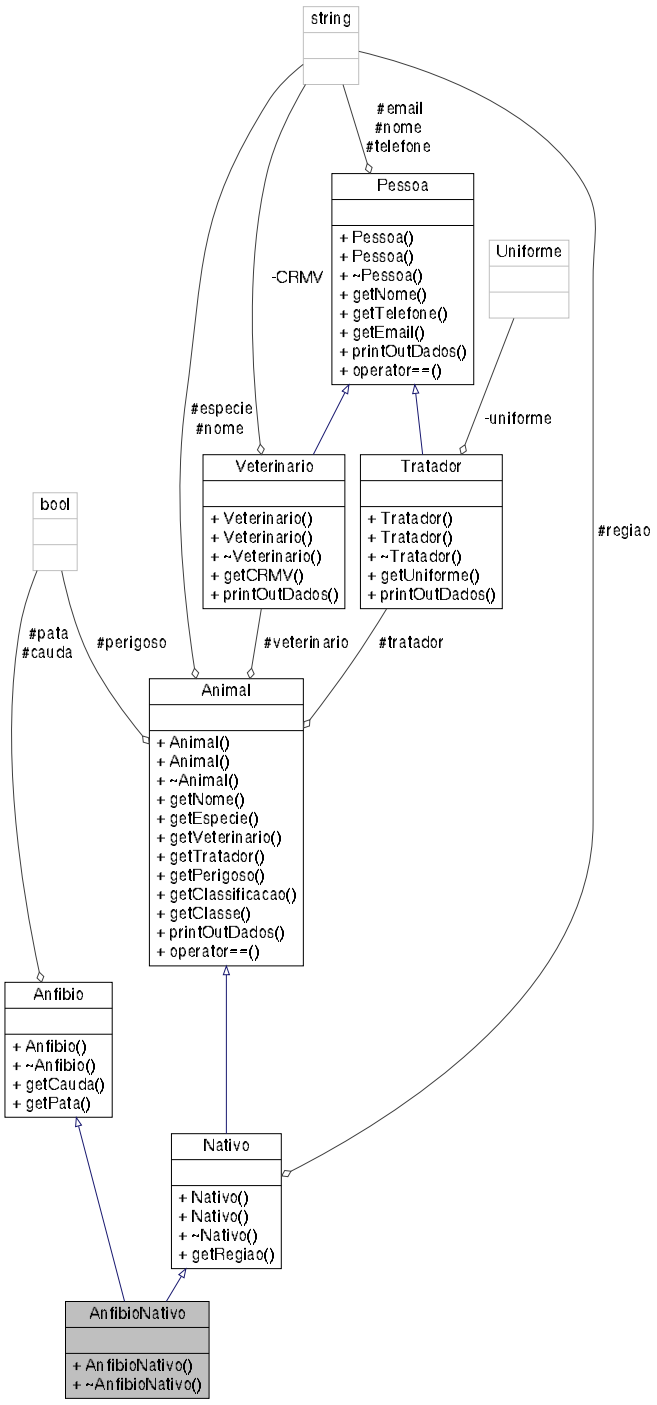


Diagrama de colaboração para AnfibioNativo:



Métodos Públicos

- **AnfibioNativo** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string regiao, bool cauda, bool pata)

Outros membros herdados

4.4.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Anfibio do tipo Nativo.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/anfibio/anfibio_nativo.hpp

4.5 Referência da Classe Animal

Implementação base para o cadastro de animais.

```
#include <animal.hpp>
```

Diagrama de Hierarquia para Animal:

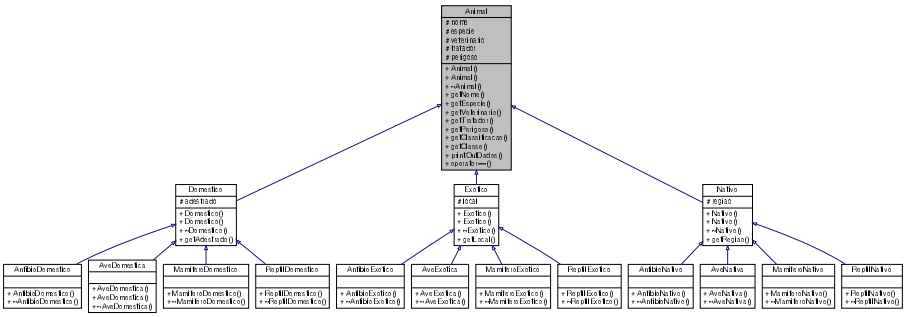
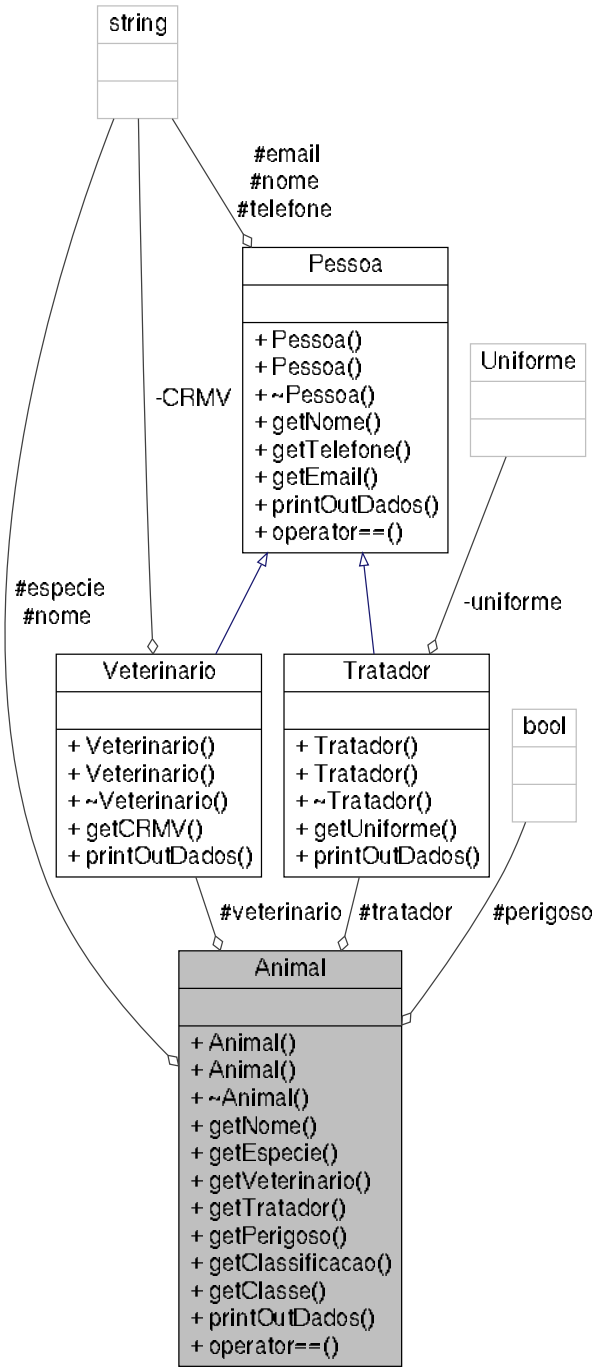


Diagrama de colaboração para Animal:



Métodos Públicos

- **Animal** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso)
- string **getNome** () const
- string **getEspecie** () const
- Veterinario **getVeterinario** () const
- Tratador **getTratador** () const

- bool **getPerigoso** () const
- string **getClassificacao** (Animal ∗animal) const
- string **getClasse** (Animal ∗animal) const
- ostream & printOutDados (ostream &o, Animal ∗animal) const

  Função para impressão de dados via sobrecarga.
- bool operator== (const Animal &outro) const

  Sobrecarga do operador de igualdade para animais.

Atributos Protegidos

- string **nome**
- string **especie**
- Veterinario **veterinario**
- Tratador tratador

  A classe do animal pode indicar a necessidade de um Tratador com Uniforme específico.
- bool **perigoso**

Amigas

- ostream & operator<< (ostream &o, Animal &animal)

  Sobrecarga do operador de extração.

4.5.1 Descrição Detalhada

Implementação base para o cadastro de animais.

O cadastro de um animal passa pela base Animal. Exigindo uma série de informações comuns a todos os animais, tais como nome, especie, seu Veterinario e Tratador.

4.5.2 Métodos

4.5.2.1 operator==()

```
bool Animal::operator== (
            const Animal & outro ) const
```

Sobrecarga do operador de igualdade para animais.

Pode ser usada para comparar a igualdade em animais, utilizando nome e especie como parâmetro para definir igualdade.

Parâmetros

Animal	Dado pela sobrecarga do operador.

Retorna

Bool confirmando (ou não) a igualdade.

4.5.2.2 printOutDados()

```
ostream& Animal::printOutDados (
            ostream & o,
            Animal * animal ) const
```

Função para impressão de dados via sobrecarga.

É chamada após o uso com operador de extração "<<". Tem sua base informando as características comuns a todos os animais, bem como sua Classe e Categoria através de checagens prévias.

Parâmetros

Animal	Dado através da sobrecarga do operador "<<".
ostream	O mesmo dado pelo operador "<<".

Retorna

Stream de saída com os dados do animal. Sendo eles os atributos definidos em Animal e a Classe e Categoria do animal.

4.5.3 Amigas e Funções Relacionadas

4.5.3.1 operator<<

```
ostream& operator<< (
            ostream & o,
            Animal & animal )  [friend]
```

Sobrecarga do operador de extração.

utilizada para chamar o método printOutDados(), podendo ser definido nas classes derivadas. Da acesso a impressão de dados do animal diretamente do stream de saída.

Retorna

Stream de saída padrão com os resultados da função printOutDados().

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/animal.hpp

4.6 Referência da Classe Ave

Classificação base para Aves.

```
#include <ave.hpp>
```

Diagrama de Hierarquia para Ave:

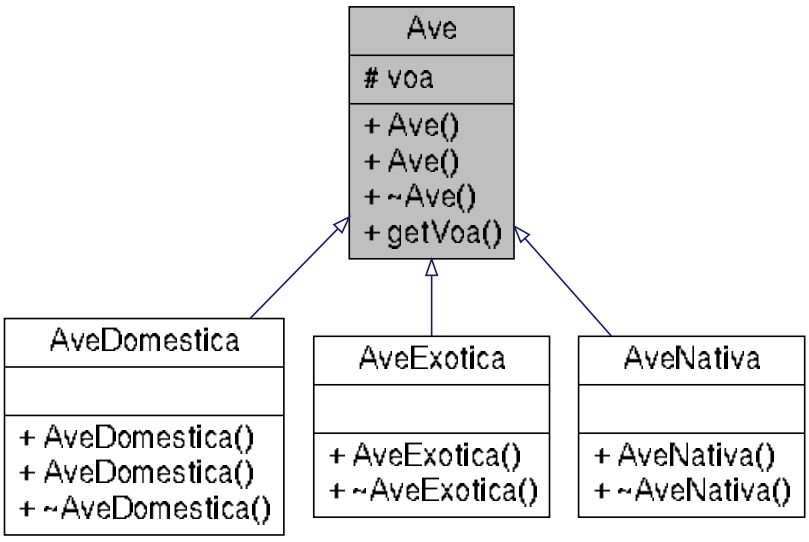
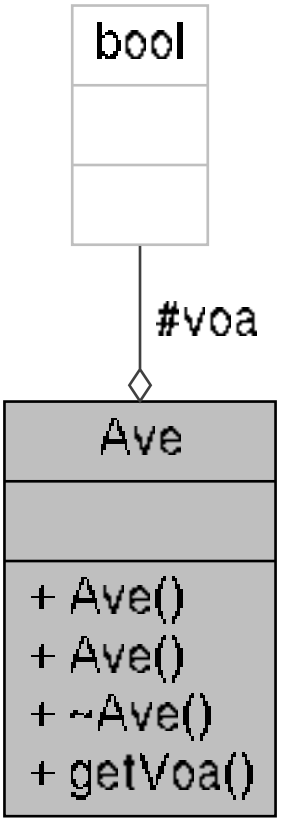


Diagrama de colaboração para Ave:



Métodos Públicos

- **Ave** (bool voa)
- bool **getVoa** () const

Atributos Protegidos

- bool **voa**

4.6.1 Descrição Detalhada

Classificação base para Aves.

A classe serve como base para os animais que se enquadram na Classe. Tendo herdeiros com base na Categoria:

- Domestico

- Nativo

- Exotico

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/ave/ave.hpp

4.7 Referência da Classe AveDomestica

Implementação de animal com Classe e Categoria.

```
#include <ave_domestica.hpp>
```

Diagrama de Hierarquia para AveDomestica:

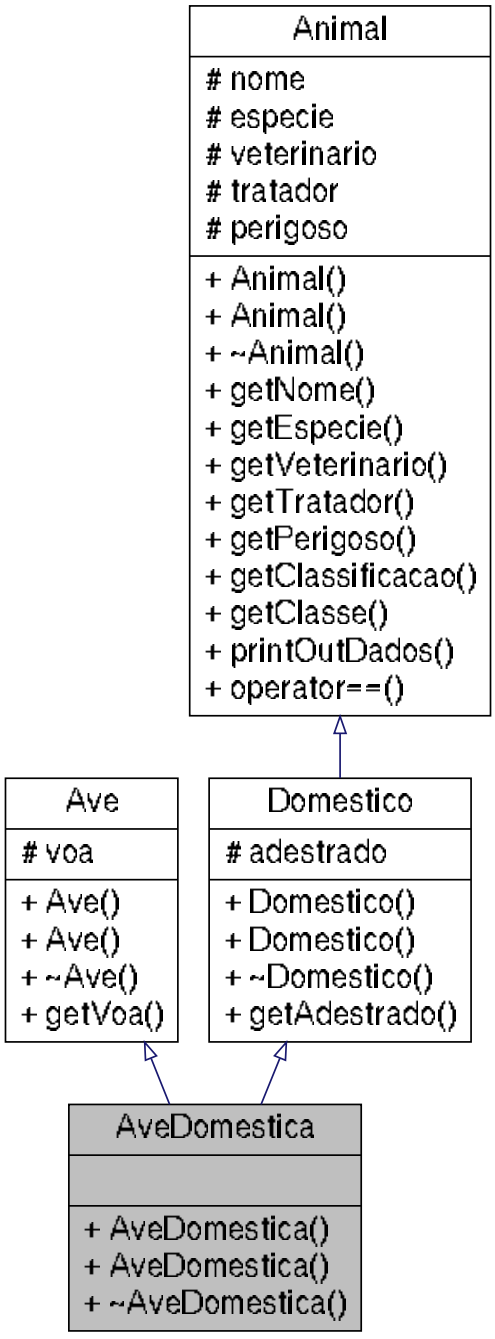
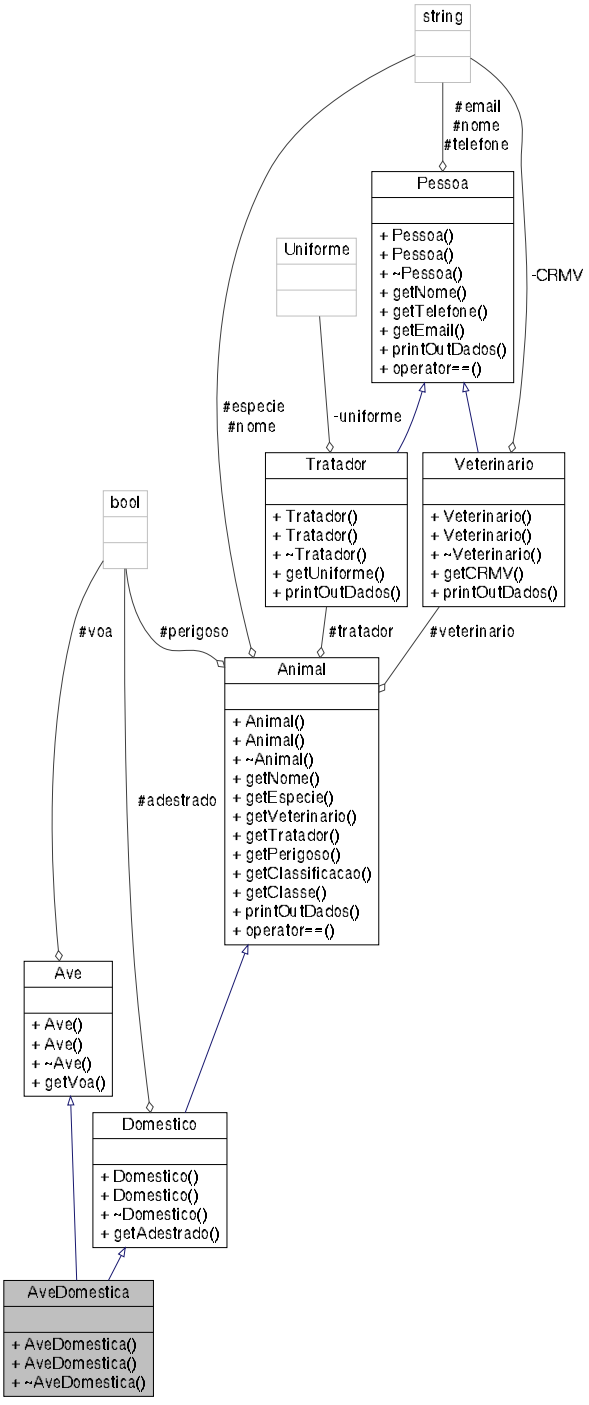


Diagrama de colaboração para AveDomestica:



Métodos Públicos

- **AveDomestica** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, bool adestrado, bool voa)

Outros membros herdados

4.7.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para uma Ave do tipo Domestico.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/ave/ave_domestica.hpp

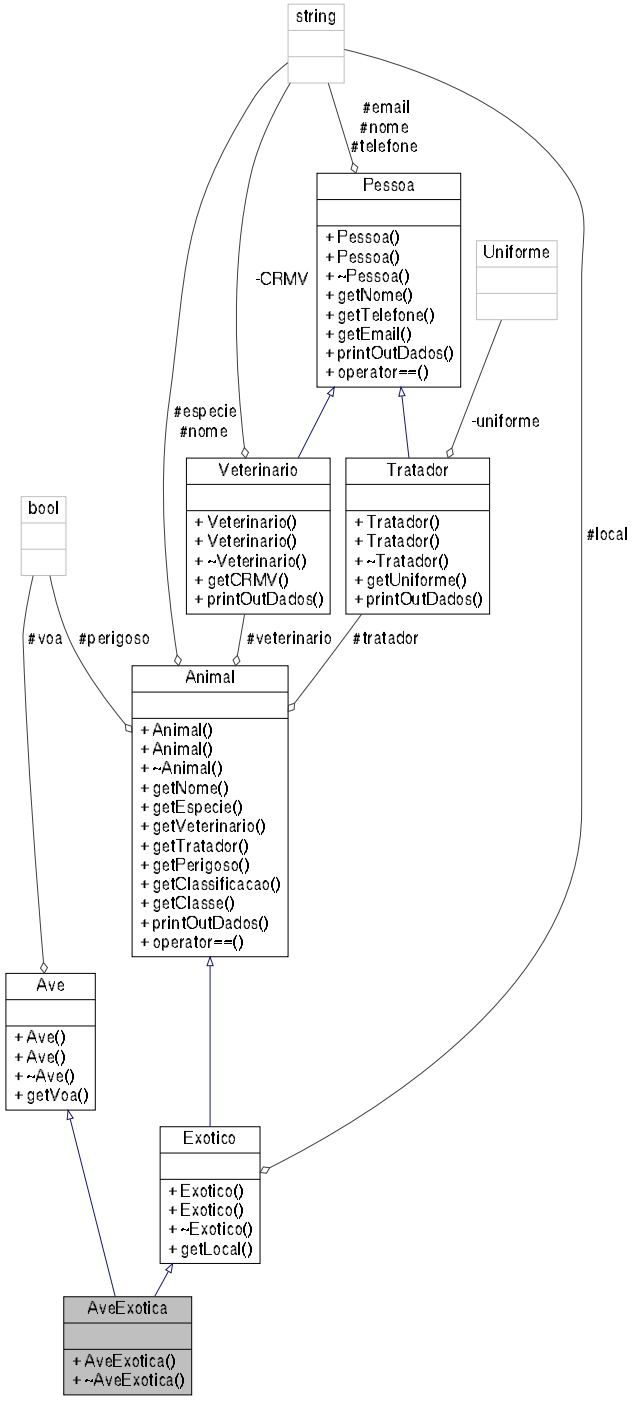
4.8 Referência da Classe AveExotica

Implementação de animal com Classe e Categoria.

```
#include <ave_exotica.hpp>
```

Diagrama de Hierarquia para AveExotica:

Diagrama de colaboração para AveExotica:



Métodos Públicos

- **AveExotica** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string local, bool voa)

Outros membros herdados

4.8.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para uma Ave do tipo Exotico.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/ave/ave_exotica.hpp

4.9 Referência da Classe AveNativa

Implementação de animal com Classe e Categoria.

```
#include <ave_nativa.hpp>
```

Diagrama de Hierarquia para AveNativa:

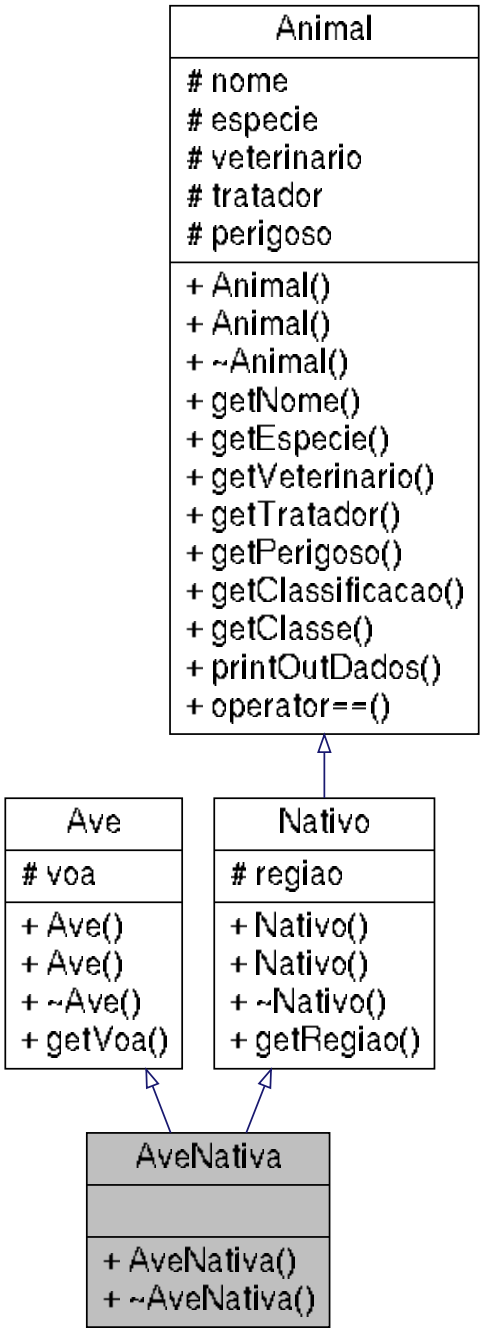
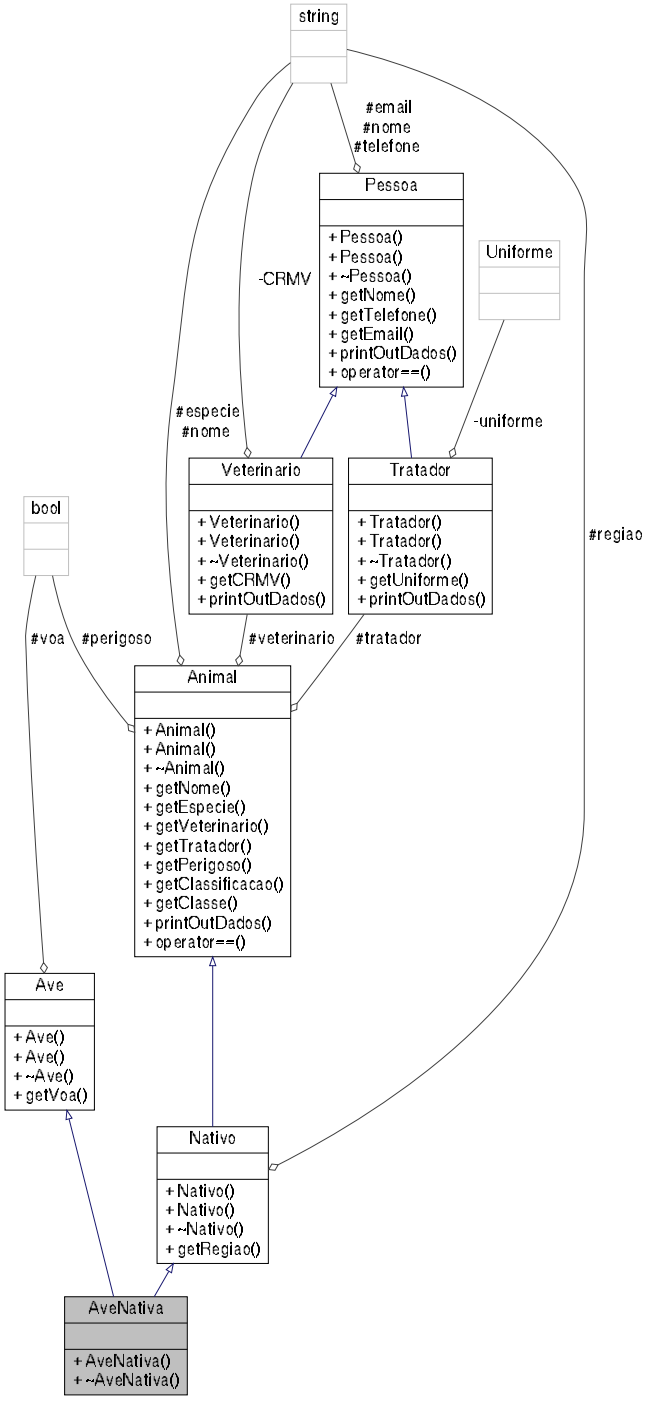


Diagrama de colaboração para AveNativa:



Métodos Públicos

- **AveNativa** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string regiao, bool voa)

Outros membros herdados

4.9.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para uma Ave do tipo Nativo.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

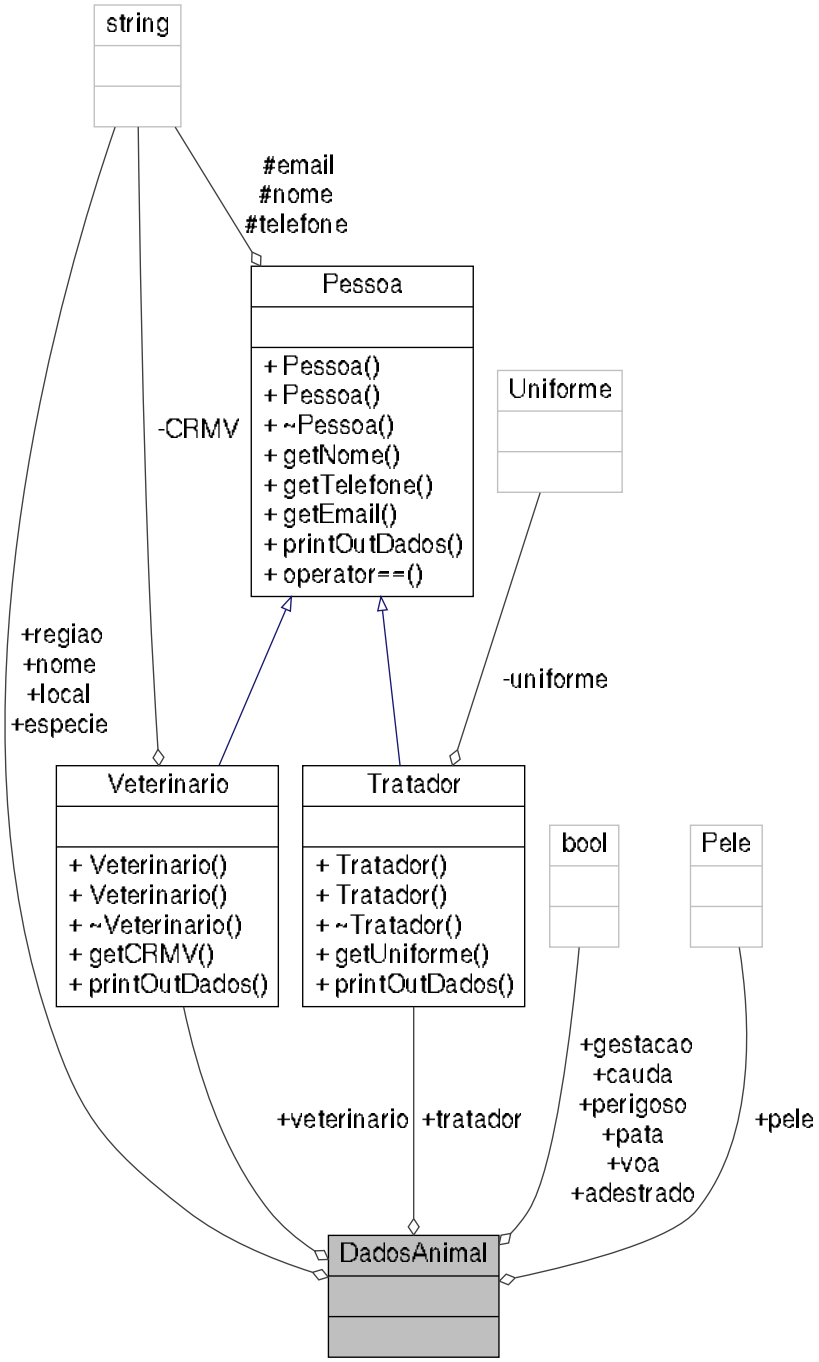
- include/animal/ave/ave_nativa.hpp

4.10 Referência da Estrutura DadosAnimal

Coringa para tipos de todos os animais.

```
#include <mapeador_animal.hpp>
```

Diagrama de colaboração para DadosAnimal:



Atributos Públicos

- string **nome**
- string **especie**
- Veterinario **veterinario**
- Tratador **tratador**
- bool **perigoso**

- string **regiao**
- string **local**
- bool **adestrado**
- bool **voa**
- bool **cauda**
- bool **pata**
- bool **gestacao**
- Pele **pele**

4.10.1 Descrição Detalhada

Coringa para tipos de todos os animais.

O struct provê uma maneira comoda de fornecer todos os atributos de animais possíveis. Sendo necessário conter as informações presentes em qualquer classe derivada de Animal já criada para o sistema. É usada na maior parte para criação de novos animais.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- include/animal/mapeador_animal.hpp

4.11 Referência da Classe Domestico

Umas das definições de categoria para Animal.

`#include <domestico.hpp>`

Diagrama de Hierarquia para Domestico:

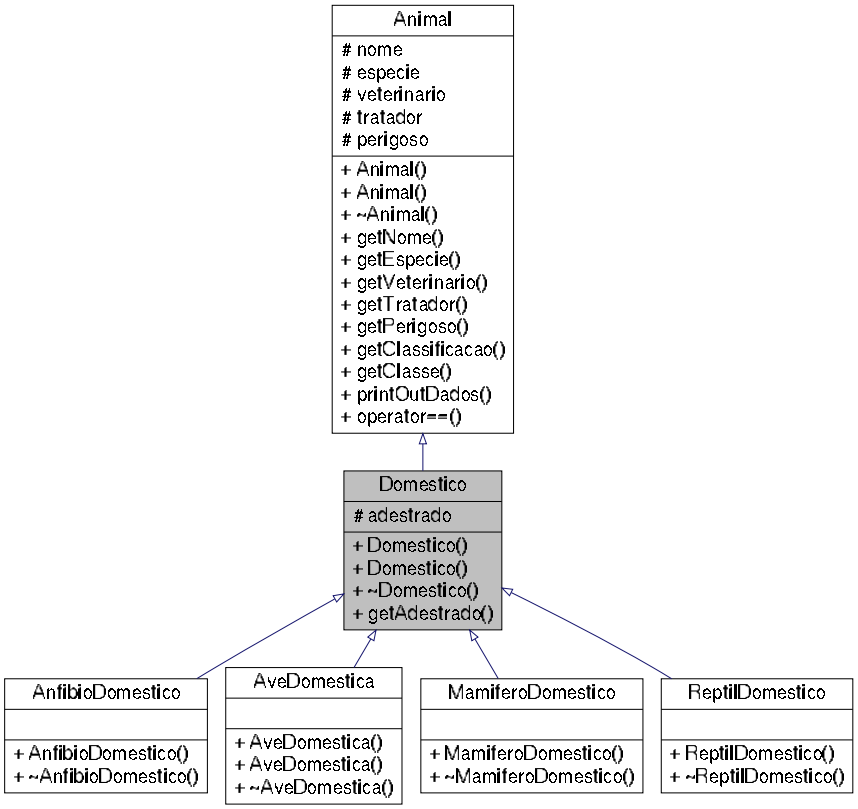
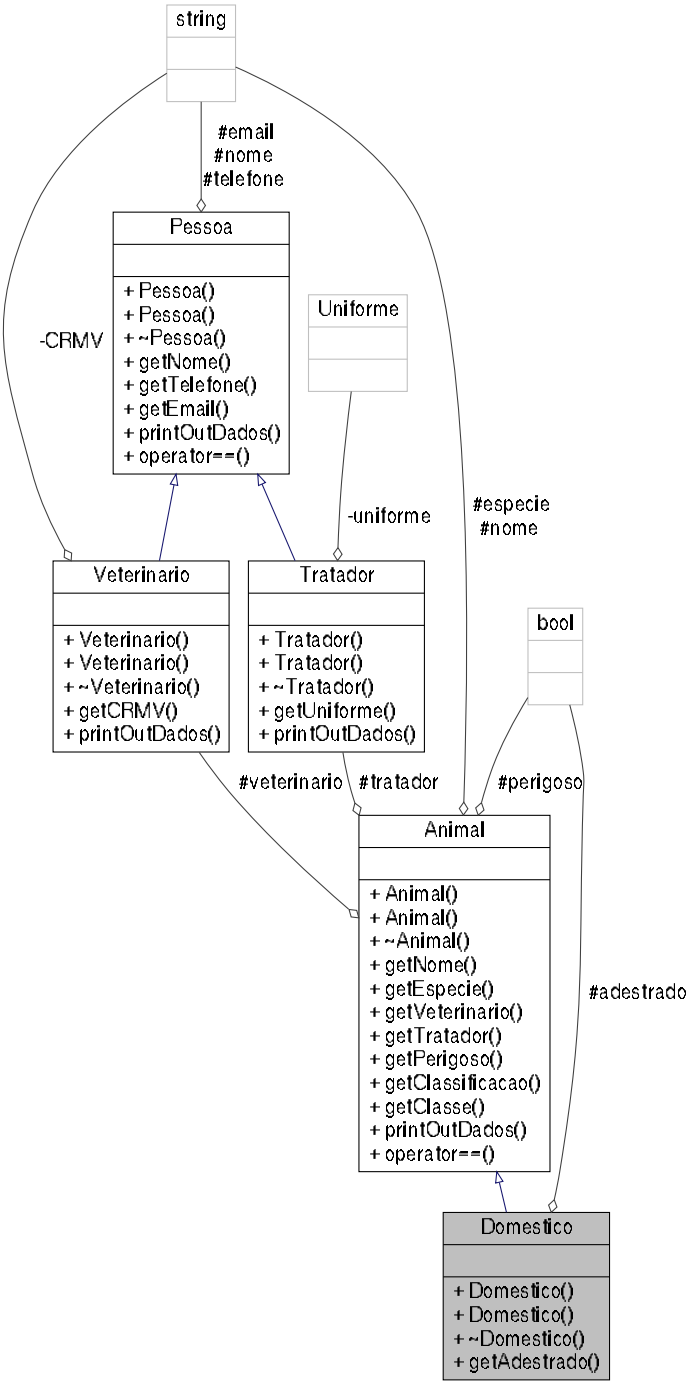


Diagrama de colaboração para Domestico:



Métodos Públicos

- **Domestico** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, bool adestrado)
- bool getAdestrado () const

  Um Domestico pode ser adestrado ou não.

Atributos Protegidos

- bool **adestrado**

4.11.1 Descrição Detalhada

Umas das definições de categoria para Animal.

Herdando animal, as classes de categoria fazem o intermédio entre a classificação do animal e as características de um animal da mesma categoria.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/domestico.hpp

4.12 Referência da Classe Exotico

Umas das definições de categoria para Animal.

```
#include <exotico.hpp>
```

Diagrama de Hierarquia para Exotico:

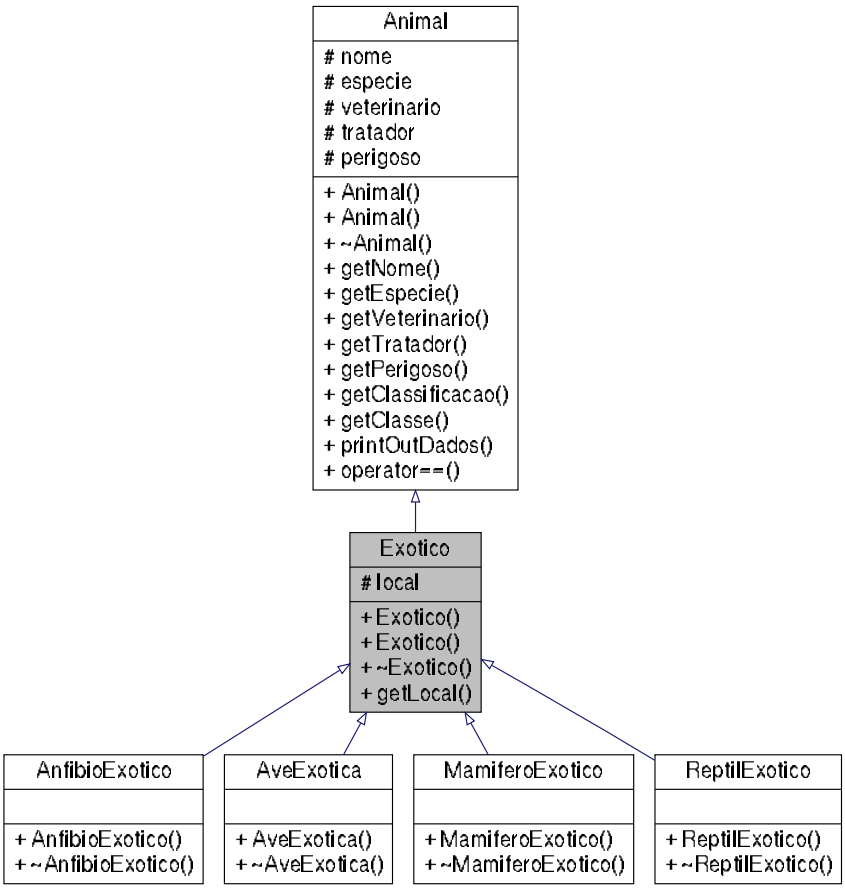
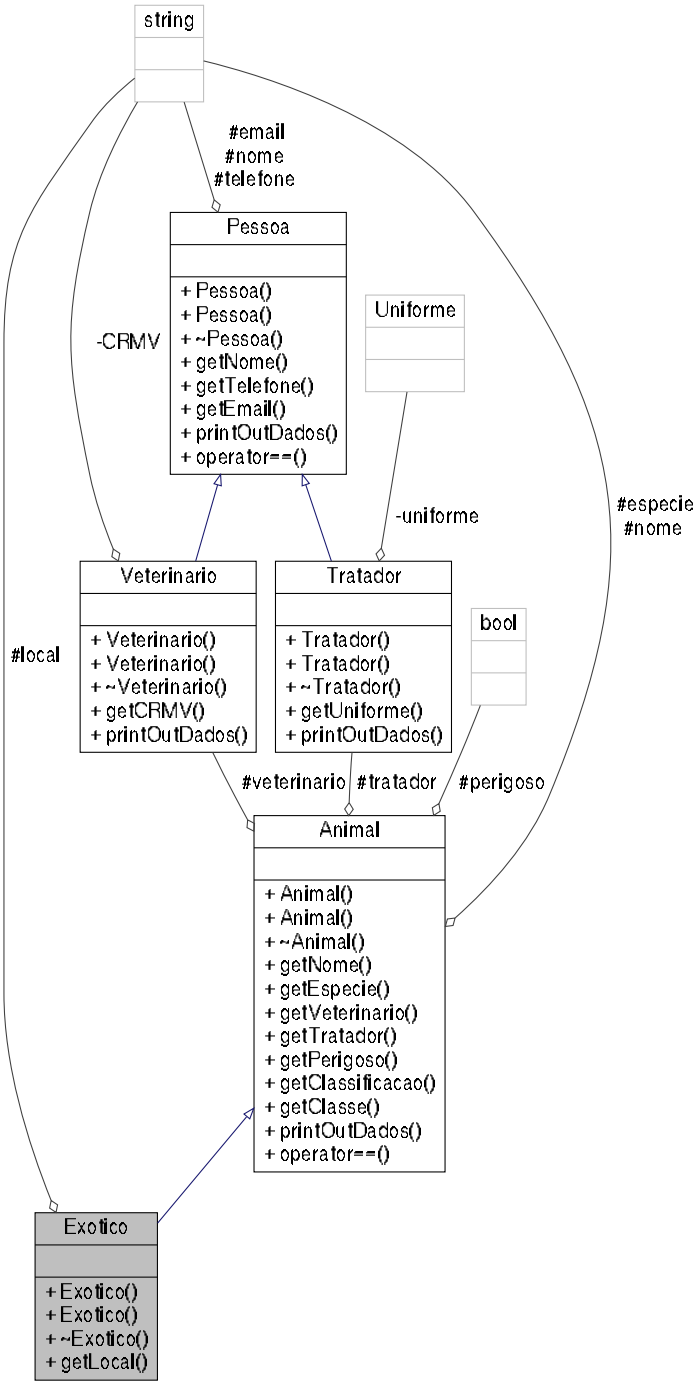


Diagrama de colaboração para Exotico:



Métodos Públicos

- **Exotico** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string local)
- string getLocal () const

  Um Exotico ter uma string de sua origem.

Atributos Protegidos

- string **local**

4.12.1 Descrição Detalhada

Umas das definições de categoria para Animal.

Herdando animal, as classes de categoria fazem o intermédio entre a classificação do animal e as características de um animal da mesma categoria.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

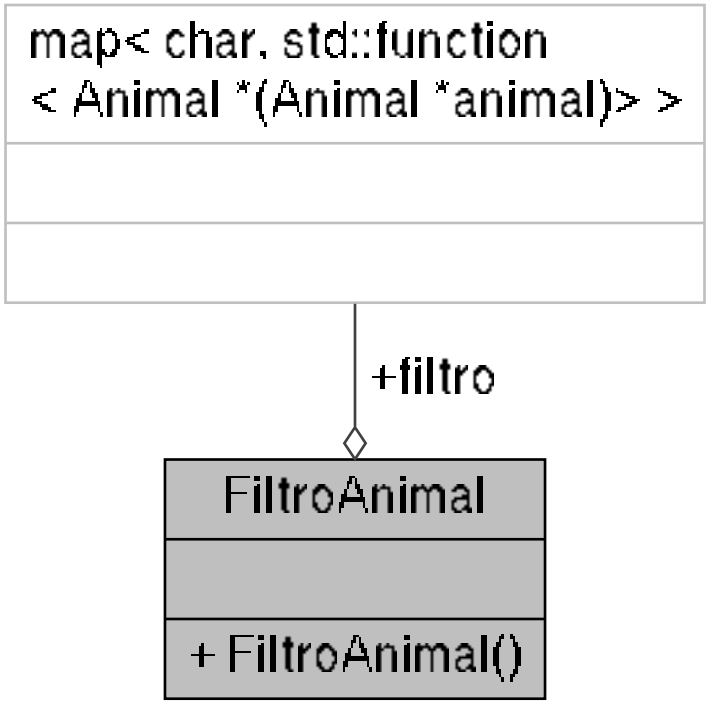
- include/animal/exotico.hpp

4.13 Referência da Classe FiltroAnimal

Classe de filtragem.

```
#include <mapeador_animal.hpp>
```

Diagrama de colaboração para FiltroAnimal:



Métodos Públicos

- FiltroAnimal ()
  Construtor do filtro.

Atributos Públicos

- std::map< char, std::function< Animal *(Animal *animal)> > filtro
  Mapa para filtragem.

4.13.1 Descrição Detalhada

Classe de filtragem.

Serve para organizar um tipo map capaz de filtrar instâncias de Animal. Retornando seu tipo em questão caso válido.

4.13.2 Construtores & Destrutores

4.13.2.1 FiltroAnimal()

```
FiltroAnimal::FiltroAnimal ( )
```

Construtor do filtro.

Inicializa os parâmetros de seu map, definindo suas opções e retornos. Sendo necessário para o uso do mesmo.@

4.13.3 Atributos

4.13.3.1 filtro

```
std::map<char, std::function<Animal* (Animal* animal)> > FiltroAnimal::filtro
```

Mapa para filtragem.

Serve como uma forma de filtrar instâncias do tipo Animal. Podendo ser usado para definir se a instância pertence a uma classificação ou categoria específica.

Parâmetros

Animal	a ser analisado.

Retorna

Referência a Animal, é nullptr caso seja ínvalido com o parâmetro dado.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/mapeador_animal.hpp

4.14 Referência da Classe Mamifero

Classificação base para Mamiferos.

```
#include <mamifero.hpp>
```

Diagrama de Hierarquia para Mamifero:

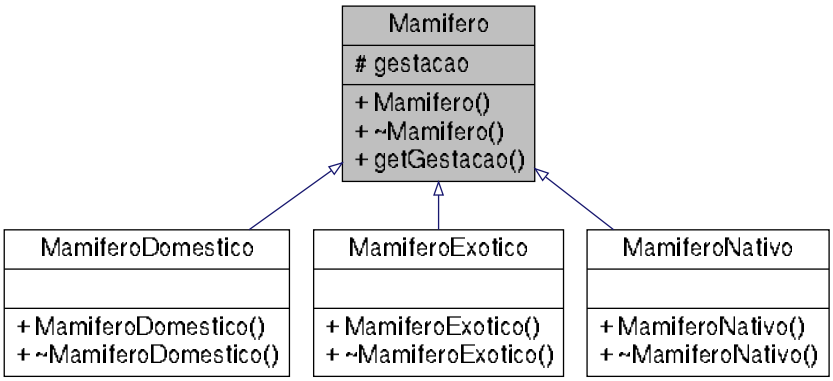
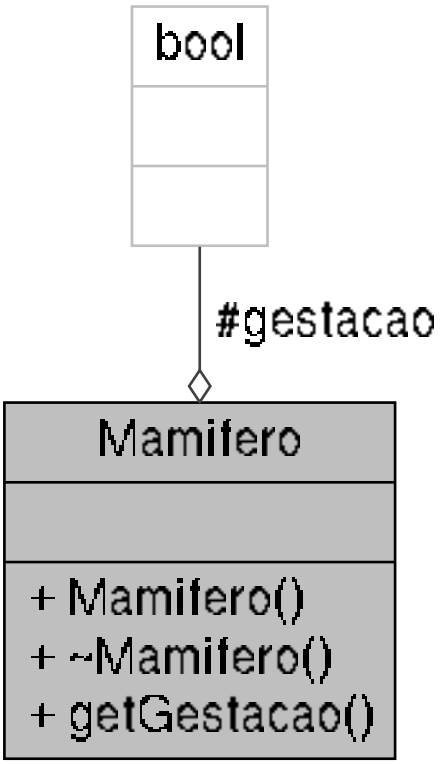


Diagrama de colaboração para Mamifero:



Métodos Públicos

- **Mamifero** (bool gestacao)
- bool **getGestacao** () const

Atributos Protegidos

- bool **gestacao**

4.14.1 Descrição Detalhada

Classificação base para Mamiferos.

A classe serve como base para os animais que se enquadram na Classe. Tendo herdeiros com base na Categoria:

- Domestico
- Nativo
- Exotico

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/mamifero/mamifero.hpp

4.15 Referência da Classe MamiferoDomestico

Implementação de animal com Classe e Categoria.

```
#include <mamifero_domestico.hpp>
```

Diagrama de Hierarquia para MamiferoDomestico:

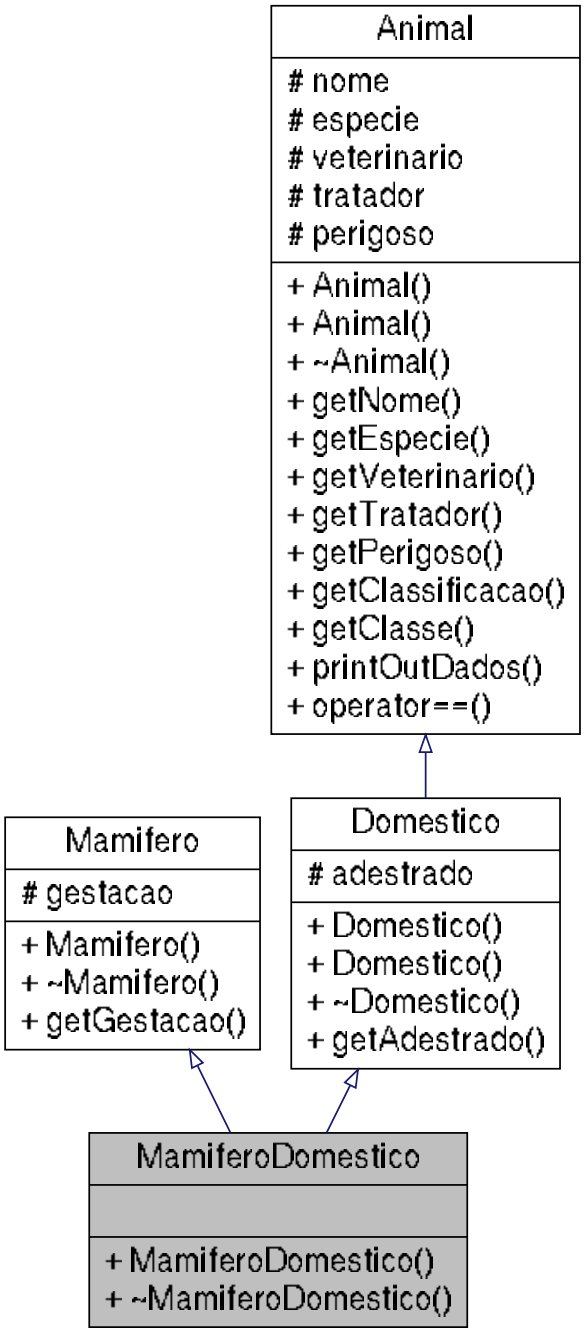
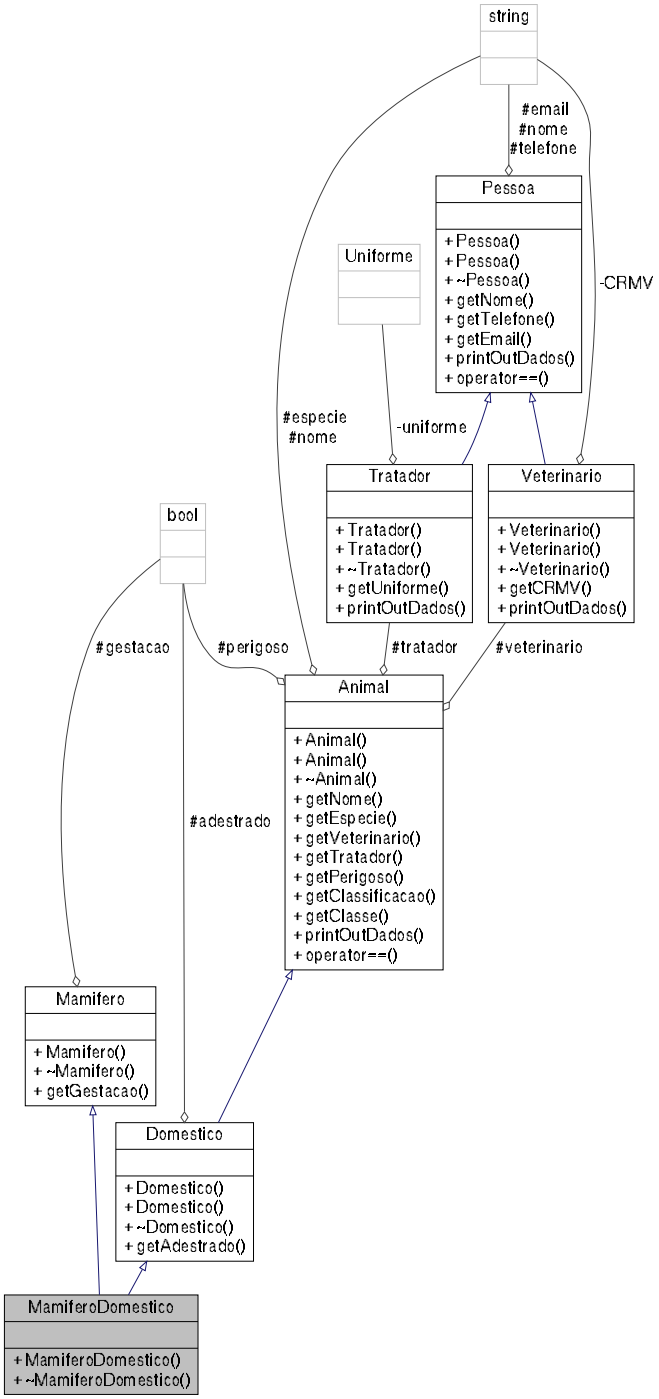


Diagrama de colaboração para MamiferoDomestico:



Métodos Públicos

- **MamiferoDomestico** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, bool adestrado, bool gestacao)

Outros membros herdados

4.15.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Mamifero do tipo Domestico.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/mamifero/mamifero_domestico.hpp

4.16 Referência da Classe MamiferoExotico

Implementação de animal com Classe e Categoria.

```
#include <mamifero_exotico.hpp>
```

Diagrama de Hierarquia para MamiferoExotico:

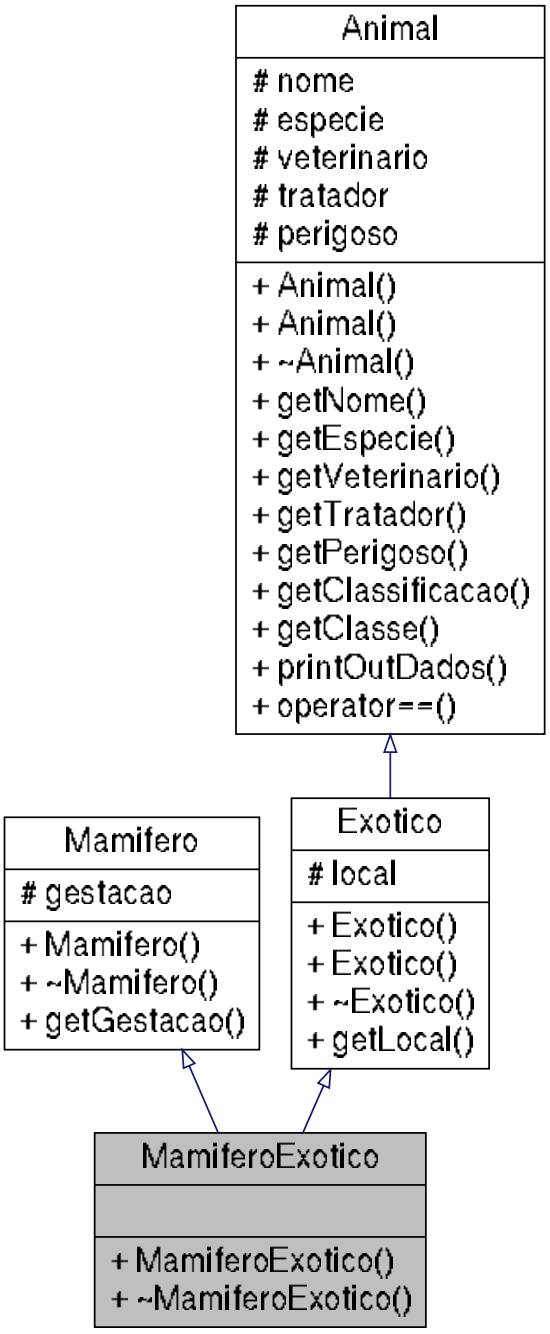
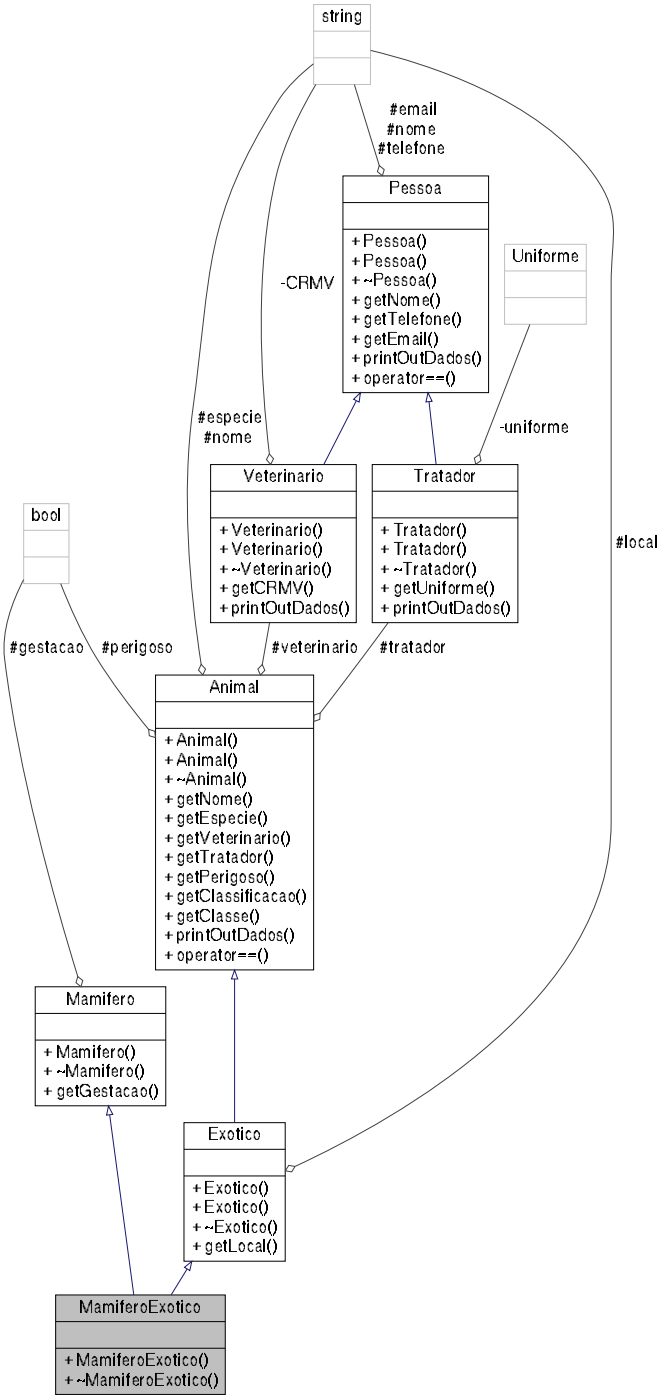


Diagrama de colaboração para MamiferoExotico:



Métodos Públicos

- **MamiferoExotico** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string local, bool gestacao)

Outros membros herdados

4.16.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Mamifero do tipo Exotico.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/mamifero/mamifero_exotico.hpp

4.17 Referência da Classe MamiferoNativo

Implementação de animal com Classe e Categoria.

```
#include <mamifero_nativo.hpp>
```

Diagrama de Hierarquia para MamiferoNativo:

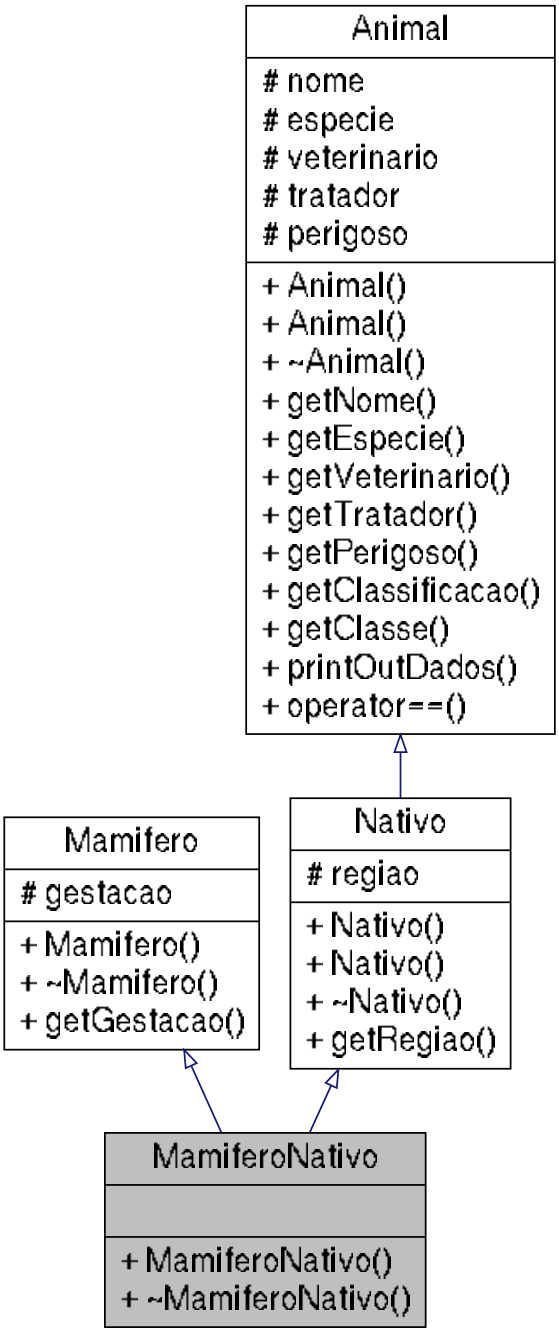
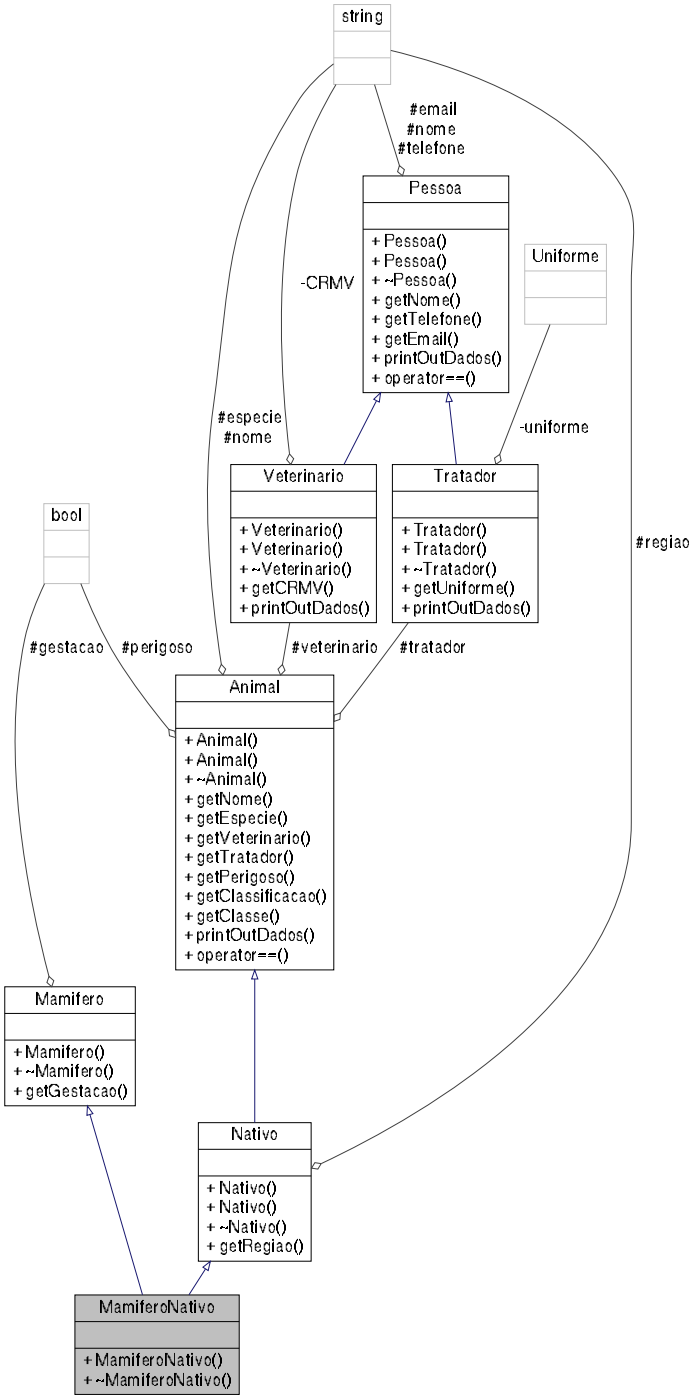


Diagrama de colaboração para MamiferoNativo:



Métodos Públicos

- **MamiferoNativo** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string regiao, bool gestacao)

Outros membros herdados

4.17.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Mamifero do tipo Nativo.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

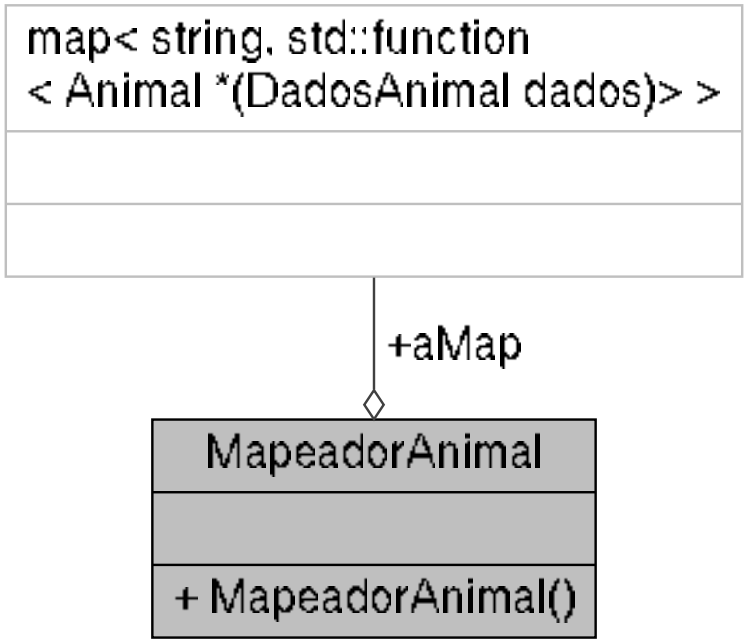
- include/animal/mamifero/mamifero_nativo.hpp

4.18 Referência da Classe MapeadorAnimal

Mapeador de animais.

`#include <mapeador_animal.hpp>`

Diagrama de colaboração para MapeadorAnimal:



Métodos Públicos

- MapeadorAnimal ()

  Construtor do mapa.

Atributos Públicos

- std::map< string, std::function< Animal *(DadosAnimal dados)> > aMap

  Mapa de animais.

4.18.1 Descrição Detalhada

Mapeador de animais.

A classe serve para conter um tipo map capaz de retornar um método de criação de um respectivo animal. A função guarda o mapa que funciona recebendo um parâmetro em string descrevendo qual animal deve ser instânciando. Então, seu retorno é justamente uma referência para o tipo específicado, utilizando funções Lambda e std::function.

4.18.2 Construtores & Destrutores

4.18.2.1 MapeadorAnimal()

```
MapeadorAnimal::MapeadorAnimal ( )
```

Construtor do mapa.

A sua importância é devido a necessidade de declarar cada caso para o seu mapa. Definindo os parâmetros e suas respostas em base a qual animal deve ser instânciado.

4.18.3 Atributos

4.18.3.1 aMap

```
std::map<string, std::function<Animal* (DadosAnimal dados)> > MapeadorAnimal::aMap
```

Mapa de animais.

Seu funcionamento ocorre pela junção de std::map e funções Lambda, no caso std::function. A sua utilidade em instanciar classes economiza código, evitando repetições do mesmo segmento e possibilitando o instânciamento de tipos derivados de Animal com apenas uma opção. Para o seu uso deve informar o tipo específico como string e os dados do animal, utilizando o struct DadosAnimal.

Parâmetros

DadosAnimal	

Retorna

Instancia para a classe desejada.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

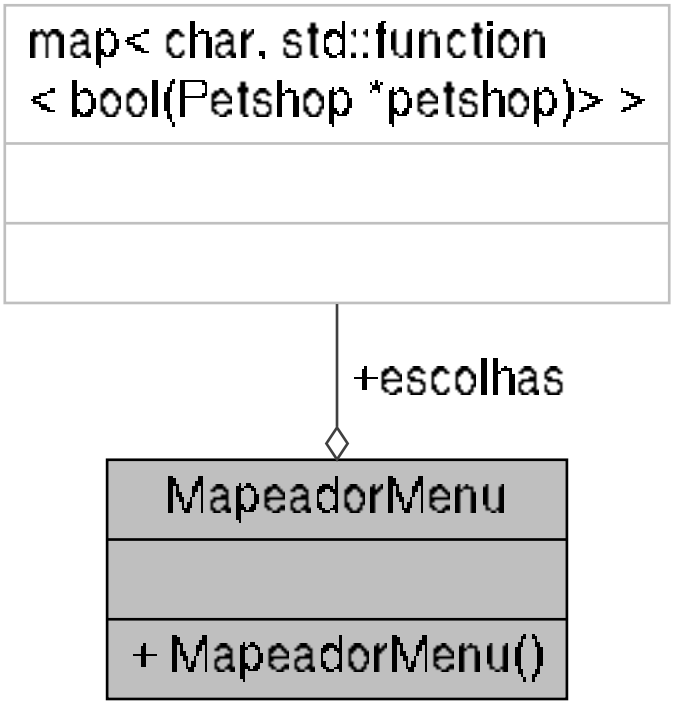
- include/animal/mapeador_animal.hpp

4.19 Referência da Classe MapeadorMenu

Classe mapeadora de funções para o menu.

`#include <mapeador_menu.hpp>`

Diagrama de colaboração para MapeadorMenu:



Métodos Públicos

- MapeadorMenu ()

  Construtor do mapeador.

Atributos Públicos

- std::map< char, std::function< bool(Petshop *petshop)> > escolhas

  Mapa usado para retornar funções.

4.19.1 Descrição Detalhada

Classe mapeadora de funções para o menu.

A classe guarda um tipo map usado para o mapeamento de opções do Menu do programa, sendo usada na função main. A sua conveniencia de juntar diversas opções e chamadas de um tipo Petshop é enorme. Com ela podemos ter várias opções advindas do Petshop sem fazer uma cadeia de condições (com IF ou SWITCH) e podemos também sempre adicionar mais retornos caso necessário.

4.19.2 Construtores & Destrutores

4.19.2.1 MapeadorMenu()

```
MapeadorMenu::MapeadorMenu ( )
```

Construtor do mapeador.

A maior importância da declaração do construtor é definir os parâmetros para seu map, sendo definido em escolhas. Nele são construidos os parâmetros do mapa para cada tipo de retorno diferente, podendo ser qualquer método Public de Petshop.

4.19.3 Atributos

4.19.3.1 escolhas

```
std::map<char, std::function<bool (Petshop* petshop)> > MapeadorMenu::escolhas
```

Mapa usado para retornar funções.

O tipo std::map pode ser acessado por um tipo char e retornando uma função lambda. Neste caso temos o uso do std::function<>. Que ao receber uma instância de Petshop, pode retornar funções advindas do mesmo, retornando a sua condição de sucesso como bool. O uso deste artífício é bem conveniente, e pode ser extendido para diversas opções possíveis e qualquer uso disponível em Public da classe Petshop.

Retorna

Função da classe Petshop desejada.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/mapeador_menu.hpp

4.20 Referência da Classe Nativo

Umas das definições de categoria para Animal.

```
#include <nativo.hpp>
```

Diagrama de Hierarquia para Nativo:

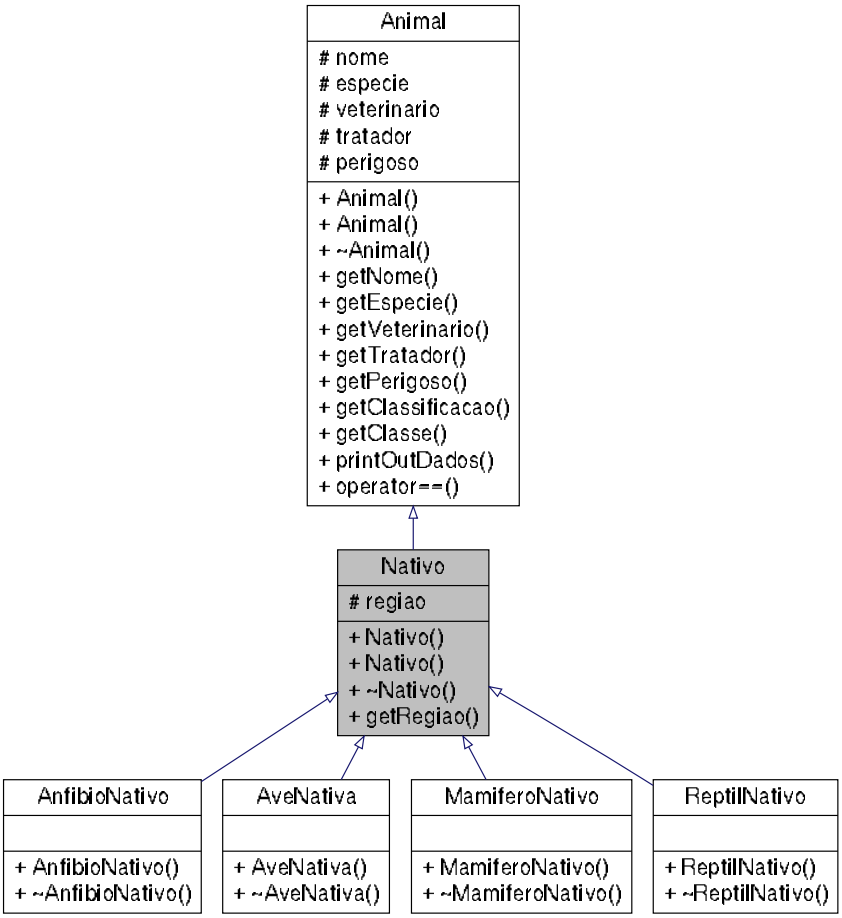
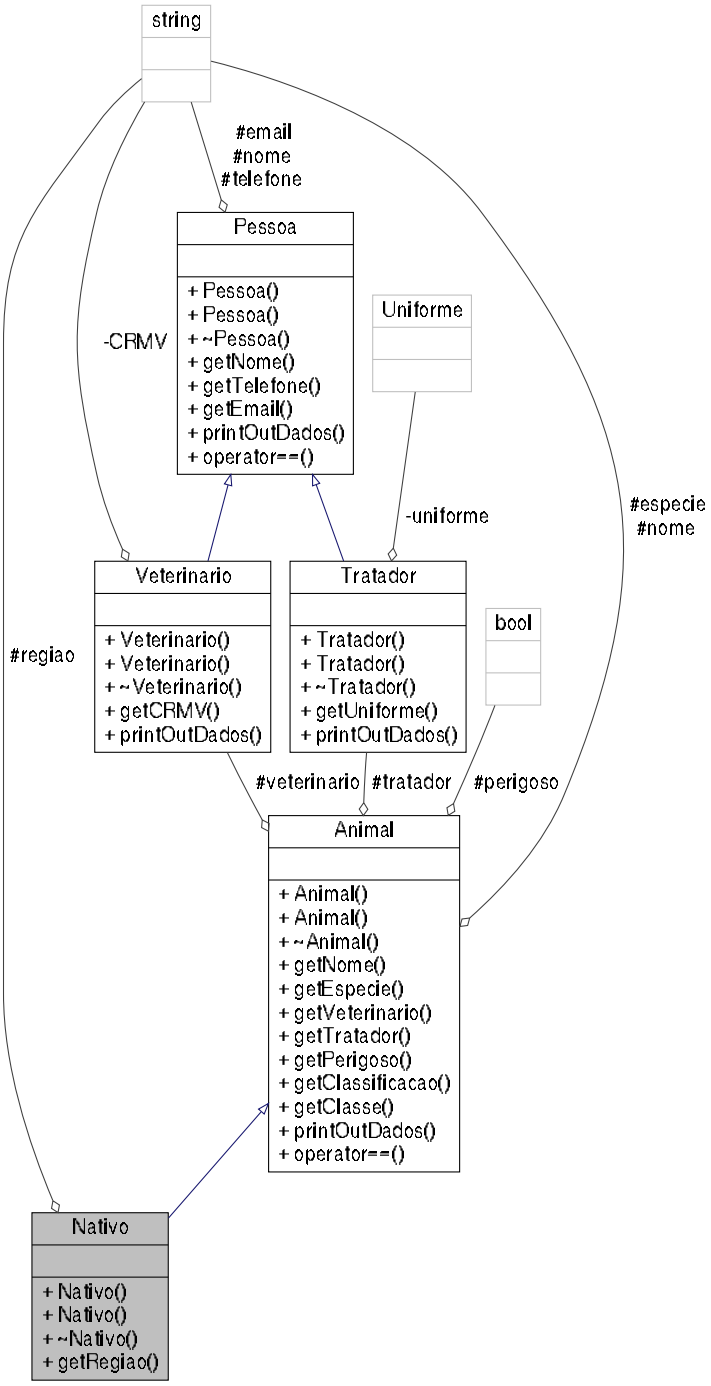


Diagrama de colaboração para Nativo:



Métodos Públicos

- **Nativo** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string regiao)
- string getRegiao () const

  Um Nativo tem a string com sua região do país.

Atributos Protegidos

- string **regiao**

4.20.1 Descrição Detalhada

Umas das definições de categoria para Animal.

Herdando animal, as classes de categoria fazem o intermédio entre a classificação do animal e as características de um animal da mesma categoria.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/nativo.hpp

4.21 Referência da Classe Pessoa

Classe base dos funcionarios.

```
#include <pessoa.hpp>
```

Diagrama de Hierarquia para Pessoa:

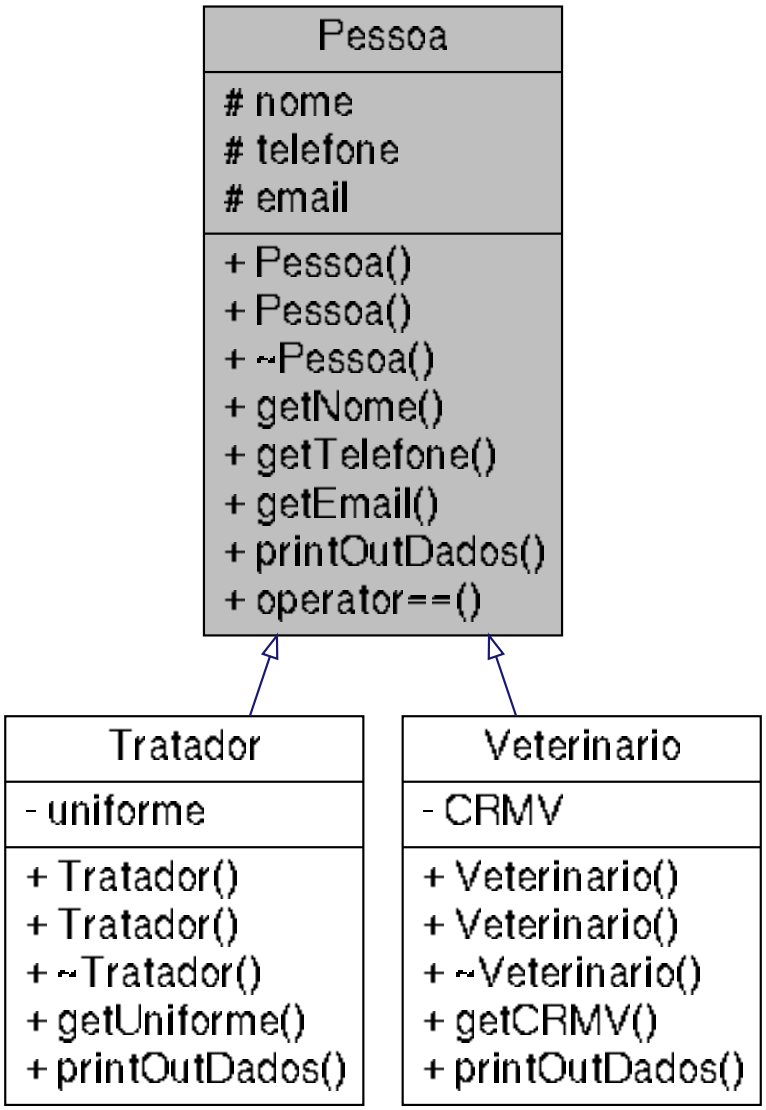
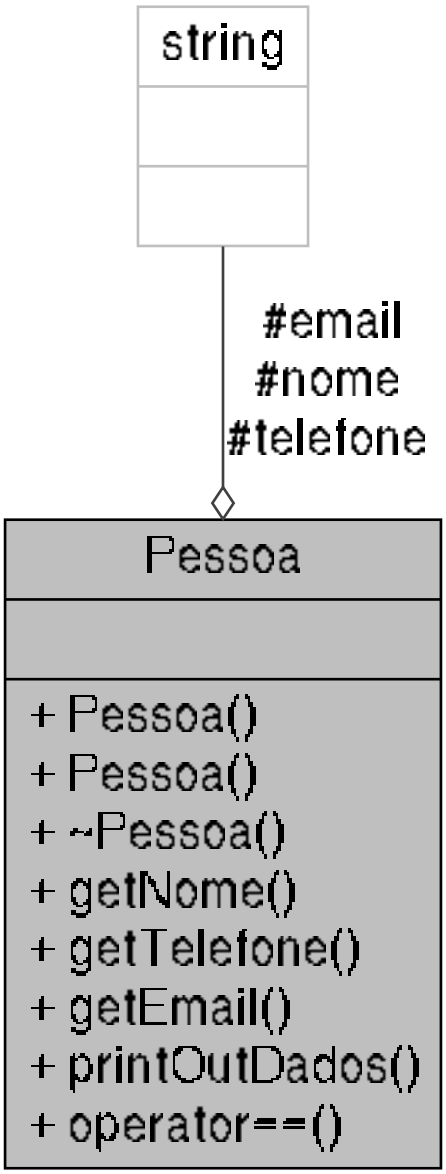


Diagrama de colaboração para Pessoa:



Métodos Públicos

- **Pessoa** (string nome, string telefone, string email)
- string **getNome** () const
- string **getTelefone** () const
- string **getEmail** () const
- virtual ostream & printOutDados (ostream &o) const =0

  Método virtual para passagem ao Cout, implementado nos herdeiros.
- bool operator== (const Pessoa &outro) const

  Sobrecarga de igualdade.

Atributos Protegidos

- string **nome**
- string **telefone**
- string **email**

Amigas

- ostream & operator<< (ostream &o, Pessoa &pessoa)

  Sobrecarga do operador de extração.

4.21.1 Descrição Detalhada

Classe base dos funcionarios.

Serve para dar base às classes de funcionarios, definindo seus atributos comuns. Sendo elas os funcionários do PetShop disponíveis a cadastro: Veterinario e Tratador.

4.21.2 Métodos

4.21.2.1 operator==()

```
bool Pessoa::operator== (
            const Pessoa & outro ) const
```

Sobrecarga de igualdade.

Parâmetros

Pessoa	sendo dado pela sobrecarga do operador.

Retorna

Bool definindo a igualdade.

4.21.2.2 printOutDados()

```
virtual ostream& Pessoa::printOutDados (
            ostream & o ) const  [pure virtual]
```

Método virtual para passagem ao Cout, implementado nos herdeiros.

Parâmetros

cout	dado pela sobrecarga na classe base.

Retorna

cout usado para impressão em stream.

Implementado por Tratador e Veterinario.

4.21.3 Amigas e Funções Relacionadas

4.21.3.1 operator<<

```
ostream& operator<< (
            ostream & o,
            Pessoa & pessoa )  [friend]
```

Sobrecarga do operador de extração.

Parâmetros

cout	dado pela operação.
Pessoa	também dado na operação.

Retorna

cout usado na impressão em stream.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

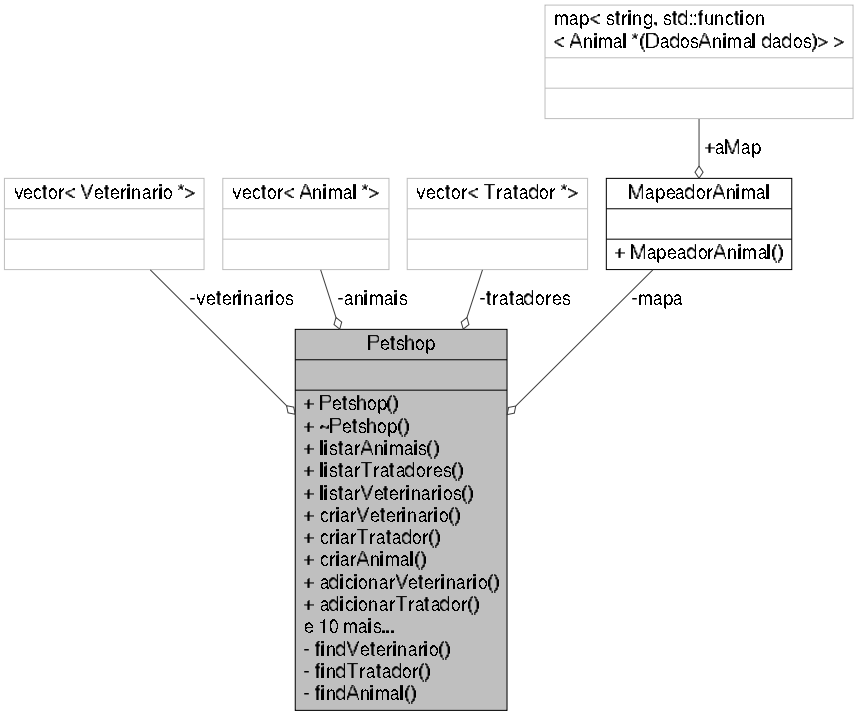
- include/funcionarios/pessoa.hpp

4.22 Referência da Classe Petshop

Classe de controle.

```
#include <petshop.hpp>
```

Diagrama de colaboração para Petshop:



Métodos Públicos

- Petshop ()

  O construtor deve ser padrão.
- ~Petshop ()

  Destrutor padrão.
- void listarAnimais ()

  Listagem de Animais no registro.
- void listarTratadores ()

  Listagem dos Tratadores registrados.
- void listarVeterinarios ()

  Listagem dos Veterinarios registrados.
- void criarVeterinario ()

  Criação de tipo Veterinario.
- void criarTratador ()

  Criação do tipo Tratador.
- void criarAnimal ()

  Criação do tipo Animal.
- bool adicionarVeterinario (Veterinario ∗vetAdd)

  Adição interna de Veterinario no sistema.
- bool adicionarTratador (Tratador ∗tratAdd)

  Adição interna de Tratador no sistema.
- bool adicionarAnimal (Animal ∗animalAdd)

  Adição interna de Animal no sistema.
- void atualizarVeterinario ()

Atualização de cadastro para Veterinario.

- void atualizarTratador ()

  Atualização de cadastro para Tratador.
- void atualizarAnimal ()

  Atualização de cadastro para Animal.
- void excluirVeterinario ()

  Exclusão de cadastro para Veterinario.
- void excluirTratador ()

  Exclusão de cadastro para Tratador.
- void excluirAnimal ()

  Exclusão de cadastro para Animal.
- Veterinario ∗ excluirVeterinario (Veterinario ∗removido)

  Remoção interna de Veterinario.
- Tratador ∗ excluirTratador (Tratador ∗removido)

  Remoção interna de Tratador.
- Animal ∗ excluirAnimal (Animal ∗removido)

  Remoção interna de Animal.

Métodos Privados

- Veterinario ∗ **findVeterinario** (string nome)
- Tratador ∗ **findTratador** (string nome)
- Animal ∗ **findAnimal** (string nome, string especie)

Atributos Privados

- vector< Veterinario ∗ > **veterinarios**
- vector< Tratador ∗ > **tratadores**
- vector< Animal ∗ > **animais**
- MapeadorAnimal **mapa**

4.22.1 Descrição Detalhada

Classe de controle.

A classe é responsável pelo controle das demais. Formando a estrutura digital do PetFera como um todo. Realizando as atividades de cadastro, administração e atualização de dados. Nela podemos:

- Adicionar, remover e atualizar dados cadastrais de...
  1. Animais
  2. Veterinarios
  3. Tratadores
- Listagem de dados de diversas origens
- Interface entre usuario
- Guardar as informações de cada classe anterior a esta.

Futuras Atividades

4.22.2 Construtores & Destrutores

4.22.2.1 Petshop()

```
Petshop::Petshop ( )
```

O construtor deve ser padrão.

Por padrão devemos ter nenhum parâmetro, sendo eles definido pela própia classe de acordo com as opções determinadas pelo usuário, de forma natural.

4.22.2.2 ~Petshop()

```
Petshop::~Petshop ( )
```

Destrutor padrão.

Tem uma função importante sendo ela a importante questão de desalocar os vetores alocados para Animal, Tratador e Veterinario.

4.22.3 Métodos

4.22.3.1 adicionarAnimal()

```
bool Petshop::adicionarAnimal (
            Animal * animalAdd )
```

Adição interna de Animal no sistema.

Realiza o processo de facto de adição no sistema para as classes do tipo Animal. Sendo chamado pelo método criarAnimal(). A função guarda a instância da classe em seu sistema, localizado no Vetor de cadastro de Animal na classe Petshop.

4.22.3.2 adicionarTratador()

```
bool Petshop::adicionarTratador (
            Tratador * tratAdd )
```

Adição interna de Tratador no sistema.

Realiza o processo de facto de adição no sistema para as classes do tipo Tratador. Sendo chamado pelo método criarTratador(). A função guarda a instância da classe em seu sistema, localizado no Vetor de cadastro de Tratador na classe Petshop.

4.22.3.3 adicionarVeterinario()

```
bool Petshop::adicionarVeterinario (
            Veterinario * vetAdd )
```

Adição interna de Veterinario no sistema.

Realiza o processo de facto de adição no sistema para as classes do tipo Veterinario. Sendo chamado pelo método criarVeterinario(). A função guarda a instância da classe em seu sistema, localizado no Vetor de cadastro de Veterinario na classe Petshop.

4.22.3.4 atualizarAnimal()

```
void Petshop::atualizarAnimal ( )
```

Atualização de cadastro para Animal.

Implementa a interface e realiza a atualização do cadastro para classes do tipo Animal no sistema. O processo pede a informação de um Animal devidamente cadastrado para o processo de atualização cadastral. É necessário prover dados semelhantes aos citados no método criarAnimal(), podendo haver uma total recriação da instancia. Para animal em específico é possível também mudar as Classificações e Categorias do animal, sendo um processo completo de atualização cadastral.

4.22.3.5 atualizarTratador()

```
void Petshop::atualizarTratador ( )
```

Atualização de cadastro para Tratador.

Implementa a interface e realiza a atualização do cadastro para classes do tipo Tratador no sistema. O processo pede a informação de um Tratador devidamente cadastrado para o processo de atualização cadastral. É necessário prover dados semelhantes aos citados no método criarTratador(), podendo haver uma total recriação da instancia.

4.22.3.6 atualizarVeterinario()

```
void Petshop::atualizarVeterinario ( )
```

Atualização de cadastro para Veterinario.

Implementa a interface e realiza a atualização do cadastro para classes do tipo Veterinario no sistema. O processo pede a informação de um Veterinario devidamente cadastrado para o processo de atualização cadastral. É necessário prover dados semelhantes aos citados no método criarVeterinario(), podendo haver uma total recriação da instancia.

4.22.3.7 criarAnimal()

```
void Petshop::criarAnimal ( )
```

Criação do tipo Animal.

Implementação da interface e função de cadastro para tipo Animal. Nesta função temos a coleta de dados para o cadastro de um novo Animal no sistema. Seus dados são postos no cadastro geral após o processo. Um animal necessita de um Veterinario e um Tratador previamente cadastrado no sistema, além disso, é necessário que o Tratador seja qualificado (pelo seu Uniforme) a trabalhar com as especificidades do Animal em questão. Para o cadastro, é necessário informar os seguintes dados:

- Nome, especie, Veterinario, Tratador
- Informar a Classificação do animal, entre elas:
  1. Ave
  2. Reptil
  3. Anfibio
  4. Mamifero
- E também sua categoria legal, podendo ser:
  1. Domestico
  2. Exotico
  3. Nativo
- Também é necessário informar específicidades de cada espécie.

4.22.3.8 criarTratador()

```
void Petshop::criarTratador ( )
```

Criação do tipo Tratador.

Implementa a interface de criação para um novo cadastro do tipo Tratador. Sendo salvo no sistema após um processo de cadastro bem sucedido. Neste método, é necessário informar as informações do Tratador a ser cadastrado, sendo elas...

- Seu nome, não podendo se repetir no sistema...
- Telefone para contato.
- E-mail
- Uniforme do mesmo, categorizando-o em um nível de segurança.

4.22.3.9 criarVeterinario()

```
void Petshop::criarVeterinario ( )
```

Criação de tipo Veterinario.

Implementa a interface de criação para um novo cadastro de Veterinario para o sistema. Após seu cadastro, o mesmo é salvo no sistema. Neste método é requirido as informações do Veterinario a ser cadastrado no sistema, como...

- Seu nome, não podendo se repetir...
- Telefone para contato.
- E-mail
- CRMV cadastrado no orgão vigente.

4.22.3.10 excluirAnimal() [1/2]

```
void Petshop::excluirAnimal ( )
```

Exclusão de cadastro para Animal.

Implementação da interface e processo de remoção de cadastro no sistema. O método após ser chamado pede por parâmetros do alvo a ser removido. Neste caso é necessário o nome e a especie do Animal a ser removido. Após o processo, o mesmo é removido do sistema e dos registros, podendo ser oferecido um ponteiro da instância do alvo removido para usos futuros. Por enquanto, a instância é apenas deletada do sistema.

4.22.3.11 excluirAnimal() [2/2]

```
Animal* Petshop::excluirAnimal (
            Animal * removido )
```

Remoção interna de Animal.

O método em questão é usado internamente pela classe. Fazendo a exclusão da instância removida do Vetor que é usado para guarda-la no sistema. A função retorna uma referência ao alvo removido, podendo ser utilizado posteriormente a quem chamou a função. Depois do método, referências sobre a instância removida não vão estar mais ao alcance da classe, pois a mesma não se encontra no Vetor organizador da classe. A função é chamada pela de mesmo nome, com tipo void, que implementa a interface chamada de excluirAnimal().

4.22.3.12 excluirTratador() [1/2]

```
void Petshop::excluirTratador ( )
```

Exclusão de cadastro para Tratador.

Implementação da interface e processo de remoção de cadastro no sistema. O método após ser chamado pede por parâmetros do alvo a ser removido. Neste caso é necessário o nome do Tratador a ser removido. Após o processo, o mesmo é removido do sistema e dos registros, podendo ser oferecido um ponteiro da instância do alvo removido para usos futuros. Por enquanto, a instância é apenas deletada do sistema.

4.22.3.13 excluirTratador() [2/2]

```
Tratador* Petshop::excluirTratador (
            Tratador * removido )
```

Remoção interna de Tratador.

O método em questão é usado internamente pela classe. Fazendo a exclusão da instância removida do Vetor que é usado para guarda-la no sistema. A função retorna uma referência ao alvo removido, podendo ser utilizado posteriormente a quem chamou a função. Depois do método, referências sobre a instância removida não vão estar mais ao alcance da classe, pois a mesma não se encontra no Vetor organizador da classe. A função é chamada pela de mesmo nome, com tipo void, que implementa a interface chamada de excluirTratador().

4.22.3.14 excluirVeterinario() [1/2]

```
void Petshop::excluirVeterinario ( )
```

Exclusão de cadastro para Veterinario.

Implementação da interface e processo de remoção de cadastro no sistema. O método após ser chamado pede por parâmetros do alvo a ser removido. Neste caso é necessário o nome do Veterinario a ser removido. Após o processo, o mesmo é removido do sistema e dos registros, podendo ser oferecido um ponteiro da instância do alvo removido para usos futuros. Por enquanto, a instância é apenas deletada do sistema.

4.22.3.15 excluirVeterinario() [2/2]

```
Veterinario* Petshop::excluirVeterinario (
            Veterinario * removido )
```

Remoção interna de Veterinario.

O método em questão é usado internamente pela classe. Fazendo a exclusão da instância removida do Vetor que é usado para guarda-la no sistema. A função retorna uma referência ao alvo removido, podendo ser utilizado posteriormente a quem chamou a função. Depois do método, referências sobre a instância removida não vão estar mais ao alcance da classe, pois a mesma não se encontra no Vetor organizador da classe. A função é chamada pela de mesmo nome, com tipo void, que implementa a interface chamada de excluirVeterinario().

4.22.3.16 listarAnimais()

```
void Petshop::listarAnimais ( )
```

Listagem de Animais no registro.

Abre a interface de listagem para animais. Abrindo as opções ao usuário de listar por filtros ou não. As opções dadas ao usuário são:

- Listagem pela classificação do animal, tais como:
  1. (A) para listar os do tipo Ave no sistema.
  2. (F) para listar os do tipo Anfibio no sistema.
  3. (R) para listar os do tipo Reptil no sistema.
  4. (M) para listar os do tipo Mamifero no sistema.

5. (F) para listar os do tipo Anfibio no sistema.

- Listagem pela categoria do IBAMA, tais como:

1. (D) listando os da categoria Domestico.
2. (E) listando os da categoria Exotico.
3. (N) listando os da categoria Nativo.

- E além disso, é possível listar animais sobre responsábilidade de um certo Veterinario ou Tratador.
- Também é possível listar todos os animais cadastrados no sistema.

4.22.3.17 listarTratadores()

```
void Petshop::listarTratadores ( )
```

Listagem dos Tratadores registrados.

Lista todos os funcionarios do tipo Tratador cadastrados no sistema. A listagem apresenta dados em relação a...

- Nome do Tratador
- Telefone para contato
- E-mail
- Uniforme do mesmo, categorizando-o em um nível de segurança.

4.22.3.18 listarVeterinarios()

```
void Petshop::listarVeterinarios ( )
```

Listagem dos Veterinarios registrados.

Lista todos os funcionarios do tipo Veterinario cadastrados no sistema. A listagem apresenta dados em relação a...

- Nome do Veterinario
- Telefone para contato
- E-mail
- CRMV cadastrado em acordo com a legislação vigente.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/petshop.hpp

4.23 Referência da Classe Reptil

Classificação base para Repteis.

```
#include <reptil.hpp>
```

Diagrama de Hierarquia para Reptil:

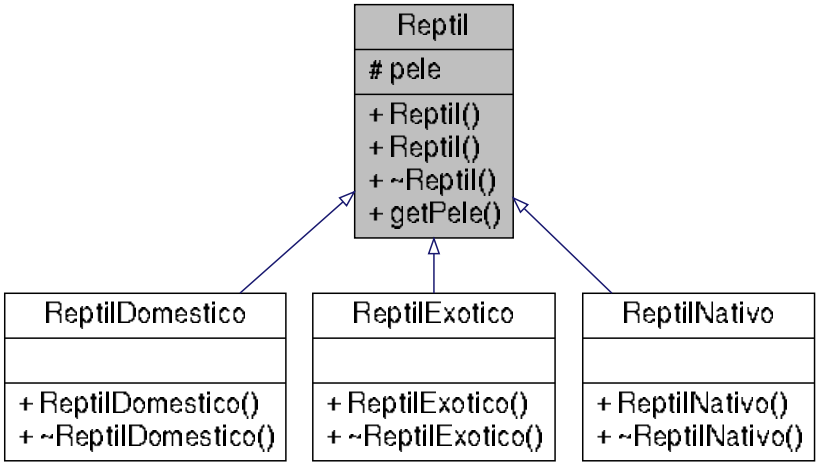
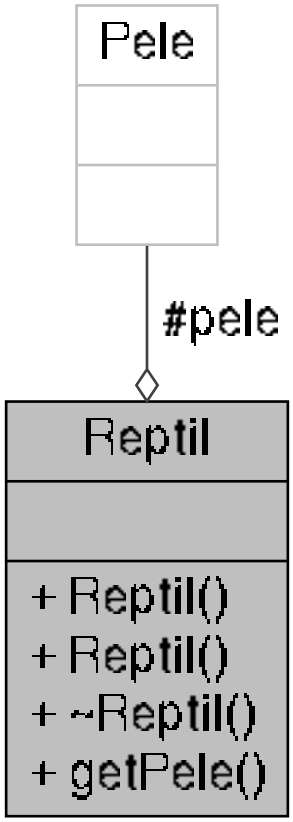


Diagrama de colaboração para Reptil:



Métodos Públicos

- **Reptil** (Pele pele)
- Pele **getPele** () const

Atributos Protegidos

- Pele **pele**

4.23.1 Descrição Detalhada

Classificação base para Repteis.

A classe serve como base para os animais que se enquadram na Classe. Tendo herdeiros com base na Categoria:

- Domestico

- Nativo

- Exotico

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/reptil/reptil.hpp

4.24 Referência da Classe ReptilDomestico

Implementação de animal com Classe e Categoria.

```
#include <reptil_domestico.hpp>
```

Diagrama de Hierarquia para ReptilDomestico:

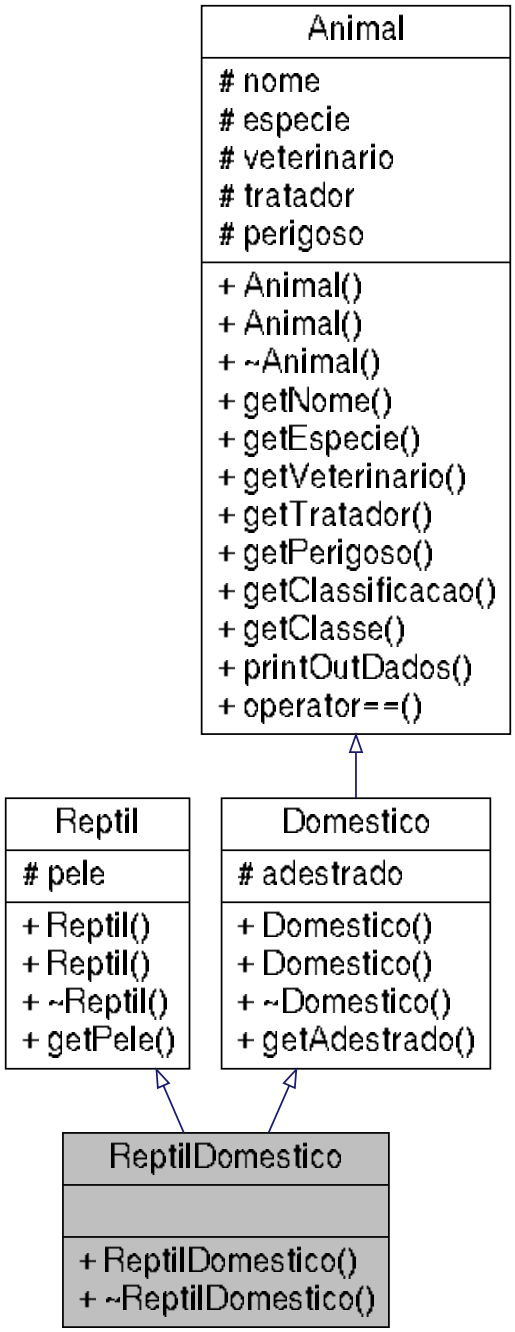
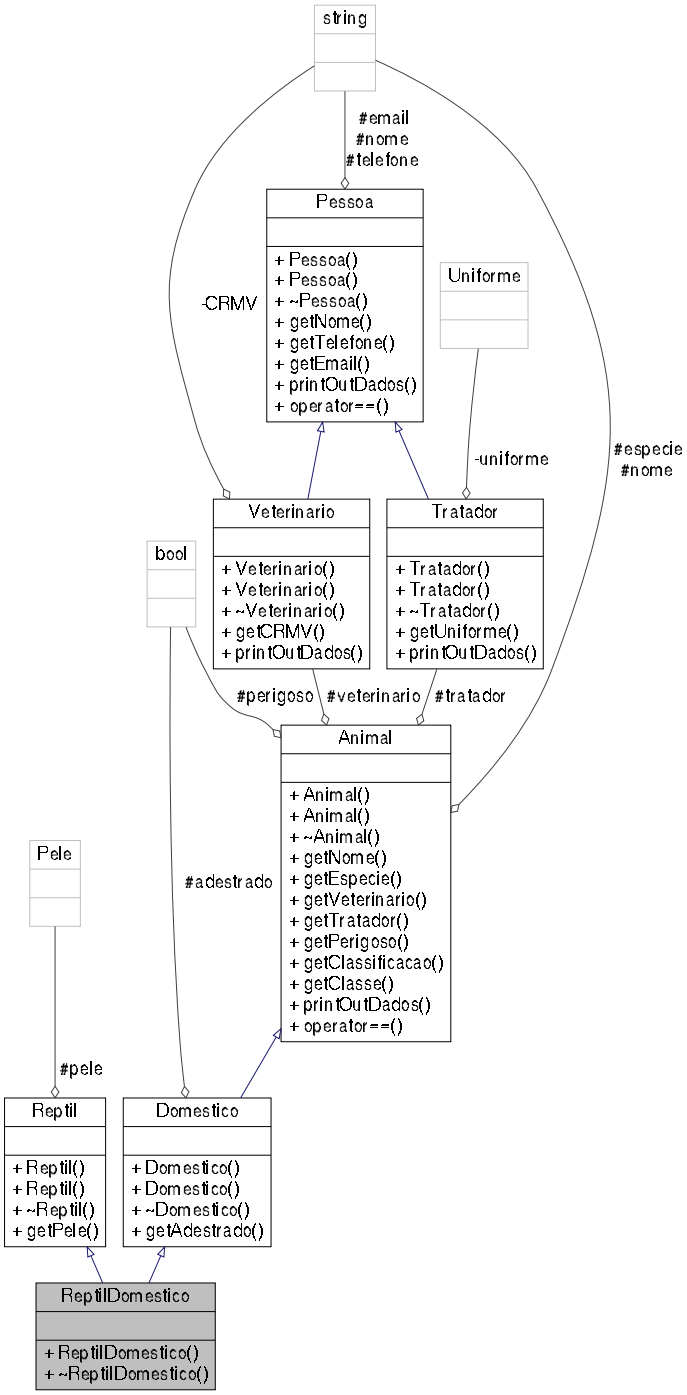


Diagrama de colaboração para ReptilDomestico:



Métodos Públicos

- **ReptilDomestico** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, bool adestrado, Pele pele)

Outros membros herdados

4.24.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Reptil do tipo Domestico.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/reptil/reptil_domestico.hpp

4.25 Referência da Classe ReptilExotico

Implementação de animal com Classe e Categoria.

```
#include <reptil_exotico.hpp>
```

Diagrama de Hierarquia para ReptilExotico:

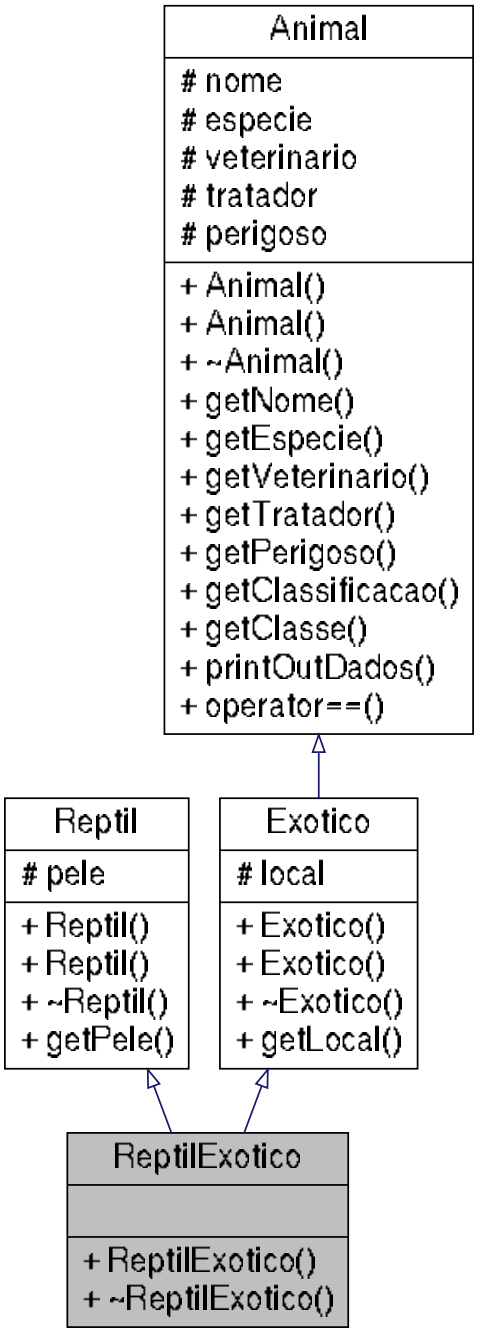
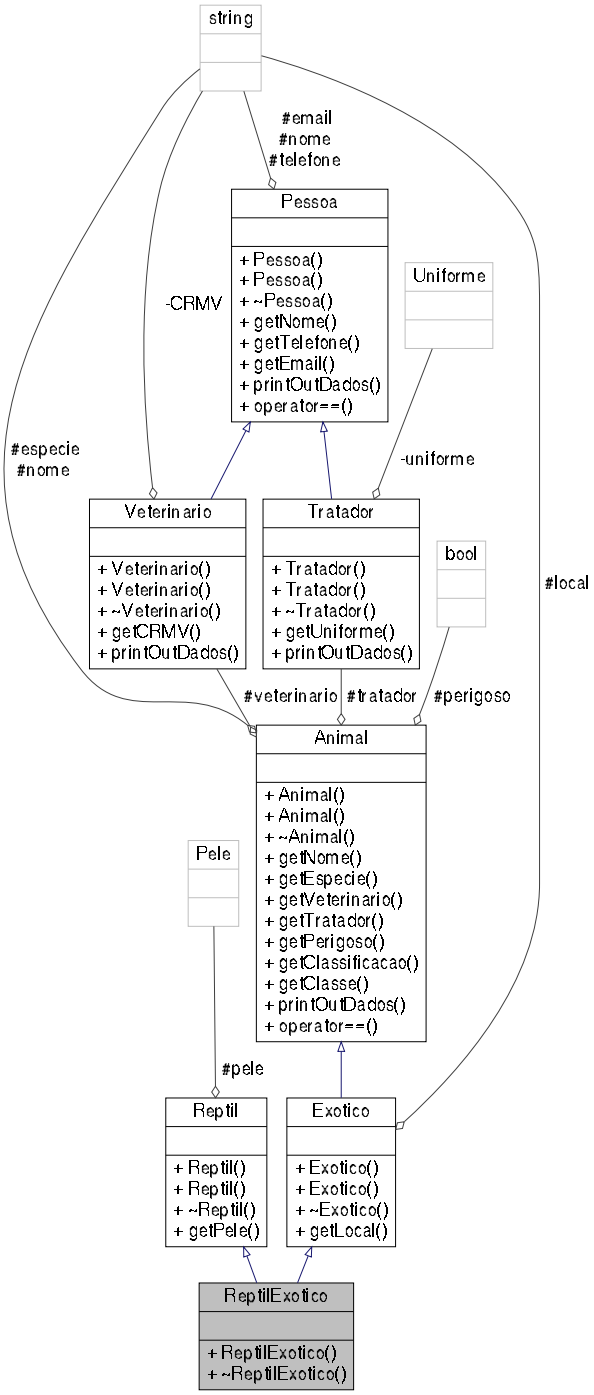


Diagrama de colaboração para ReptilExotico:



Métodos Públicos

- **ReptilExotico** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string local, Pele pele)

Outros membros herdados

4.25.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Reptil do tipo Exotico.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/reptil/reptil_exotico.hpp

4.26 Referência da Classe ReptilNativo

Implementação de animal com Classe e Categoria.

```
#include <reptil_nativo.hpp>
```

Diagrama de Hierarquia para ReptilNativo:

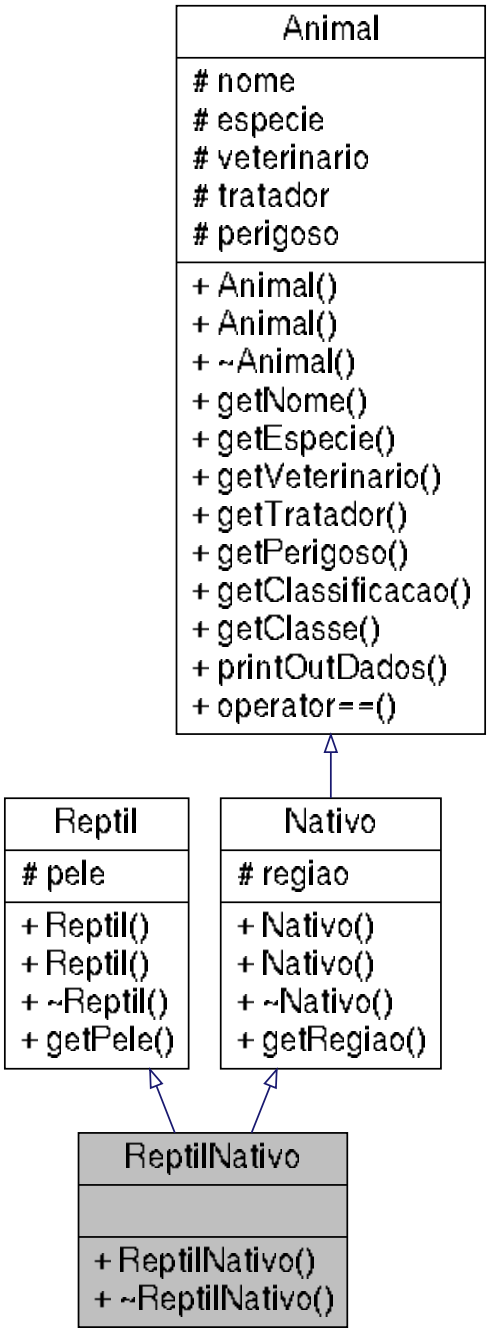
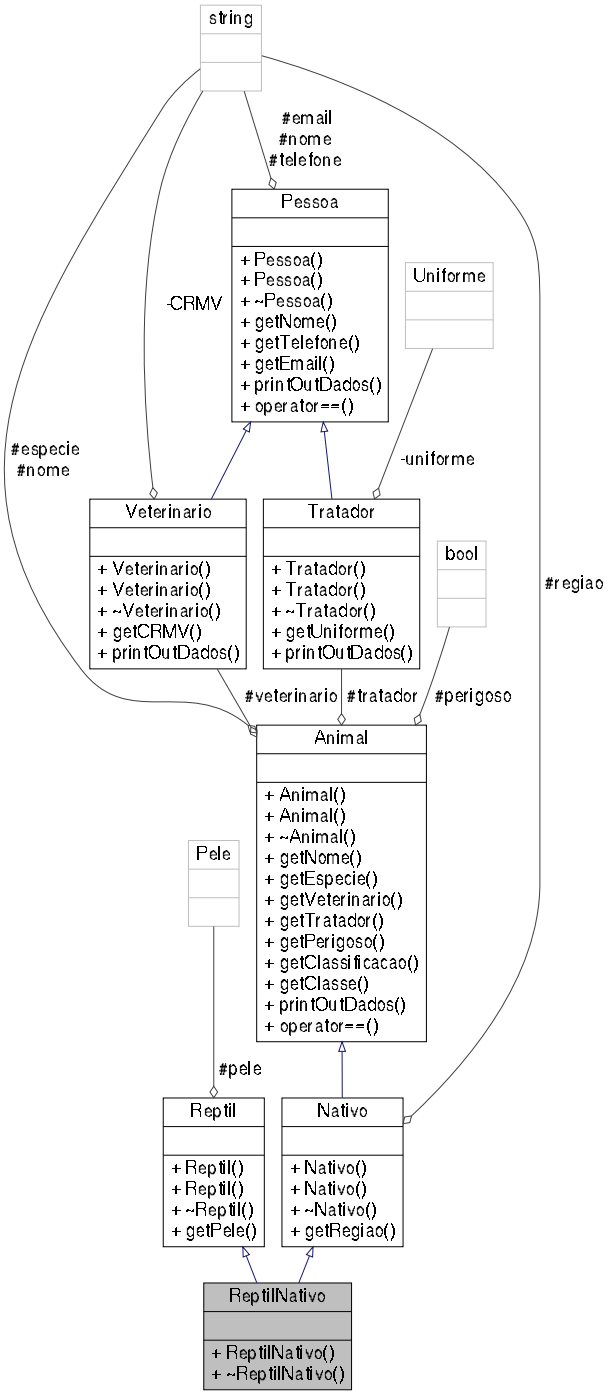


Diagrama de colaboração para ReptilNativo:



Métodos Públicos

- **ReptilNativo** (string nome, string especie, Veterinario veterinario, Tratador tratador, bool perigoso, string regiao, Pele pele)

Outros membros herdados

4.26.1 Descrição Detalhada

Implementação de animal com Classe e Categoria.

As classes finais que de fato são usadas para instanciamento e administração dos Animais devem ter esta assinatura. Possuindo um tipo que o classifique e o cateogorize. Sendo a classe do mesmo feita por herança multipla. Aqui temos uma definição para um Reptil do tipo Nativo.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/animal/reptil/reptil_nativo.hpp

4.27 Referência da Classe Tratador

Implementação dos tratadores.

```
#include <tratador.hpp>
```

Diagrama de Hierarquia para Tratador:

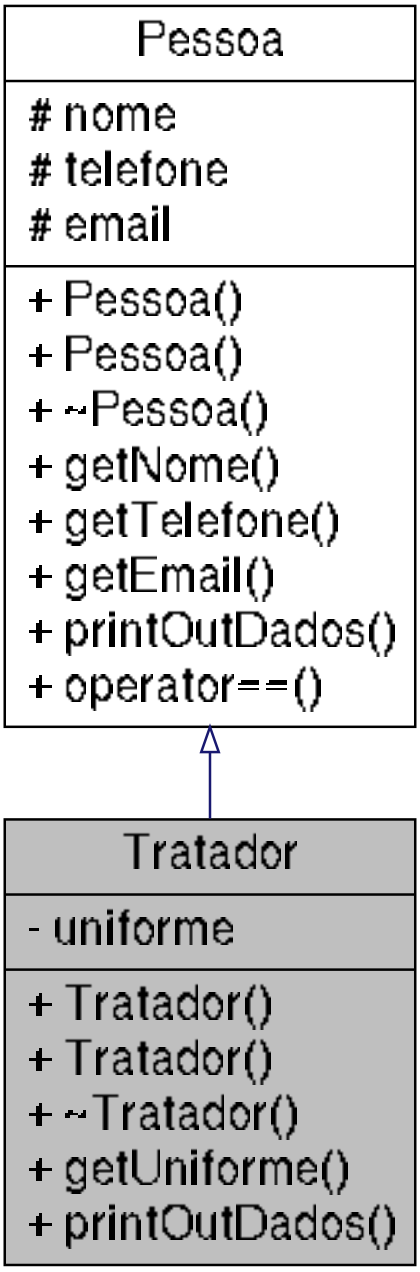
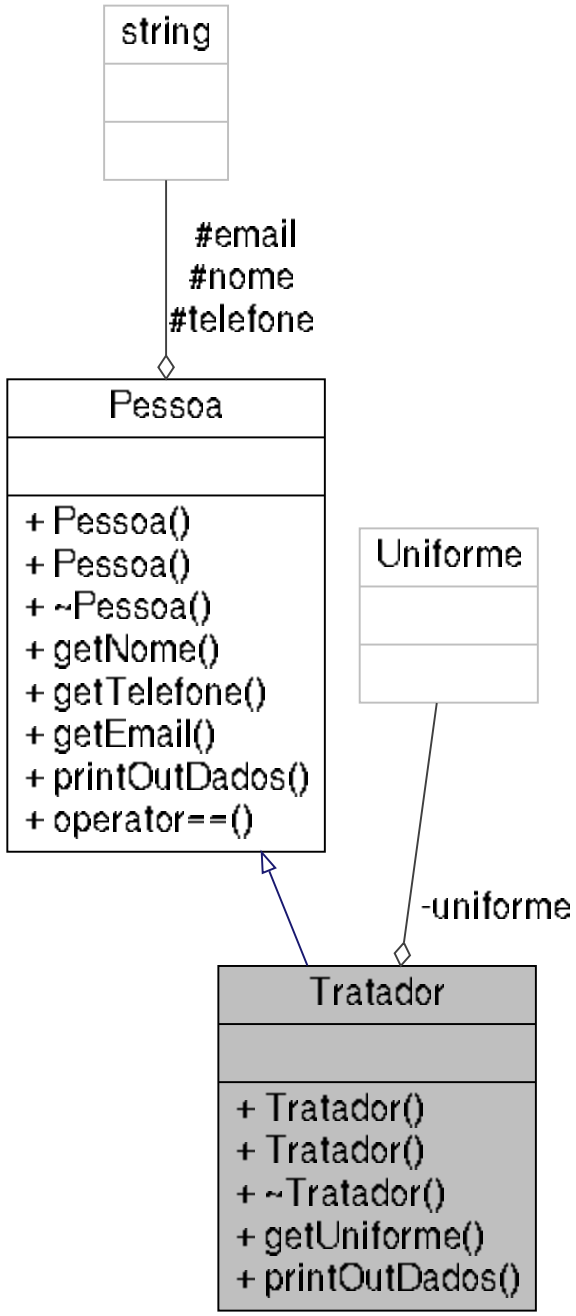


Diagrama de colaboração para Tratador:



Métodos Públicos

- **Tratador** (string nome, string telefone, string email, Uniforme uniforme)
- Uniforme **getUniforme** () const
- ostream & printOutDados (ostream &o) const

  Método virtual para passagem ao Cout, implementado nos herdeiros.

Atributos Privados

- Uniforme **uniforme**

Outros membros herdados

4.27.1 Descrição Detalhada

Implementação dos tratadores.

A critério das necessidades do PetShop... A classe deve prover as informações básicas categorizadas em Pessoa. Tão quanto as classificações em relação a segurança no que diz respeito a quais animais o tratador pode se responsabilizar, sendo definida pelo seu Uniforme.

4.27.2 Métodos

4.27.2.1 printOutDados()

```
ostream& Tratador::printOutDados (
            ostream & o ) const  [virtual]
```

Método virtual para passagem ao Cout, implementado nos herdeiros.

Parâmetros

cout	dado pela sobrecarga na classe base.

Retorna

cout usado para impressão em stream.

Implementa Pessoa.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/funcionarios/tratador.hpp

4.28 Referência da Classe Veterinario

Implementação dos veterinarios.

```
#include <veterinario.hpp>
```

Diagrama de Hierarquia para Veterinario:

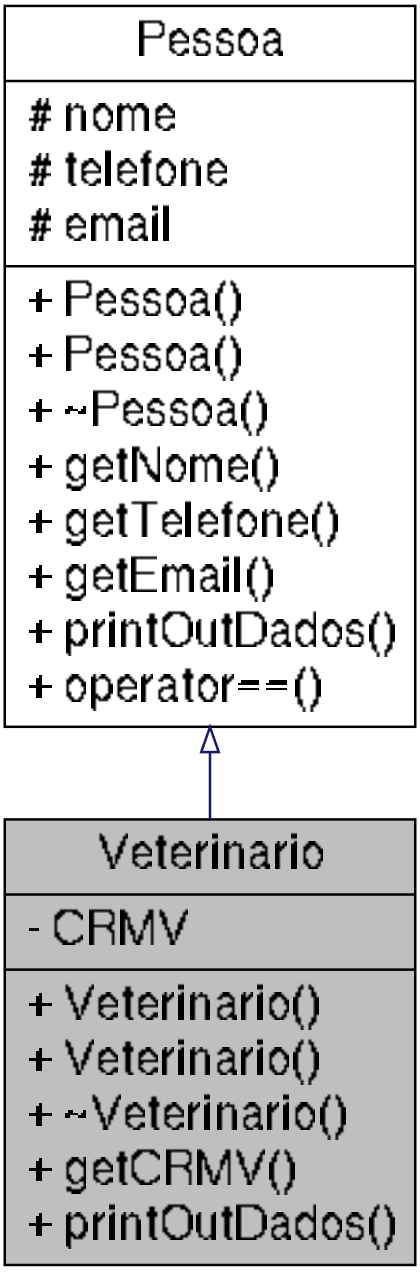
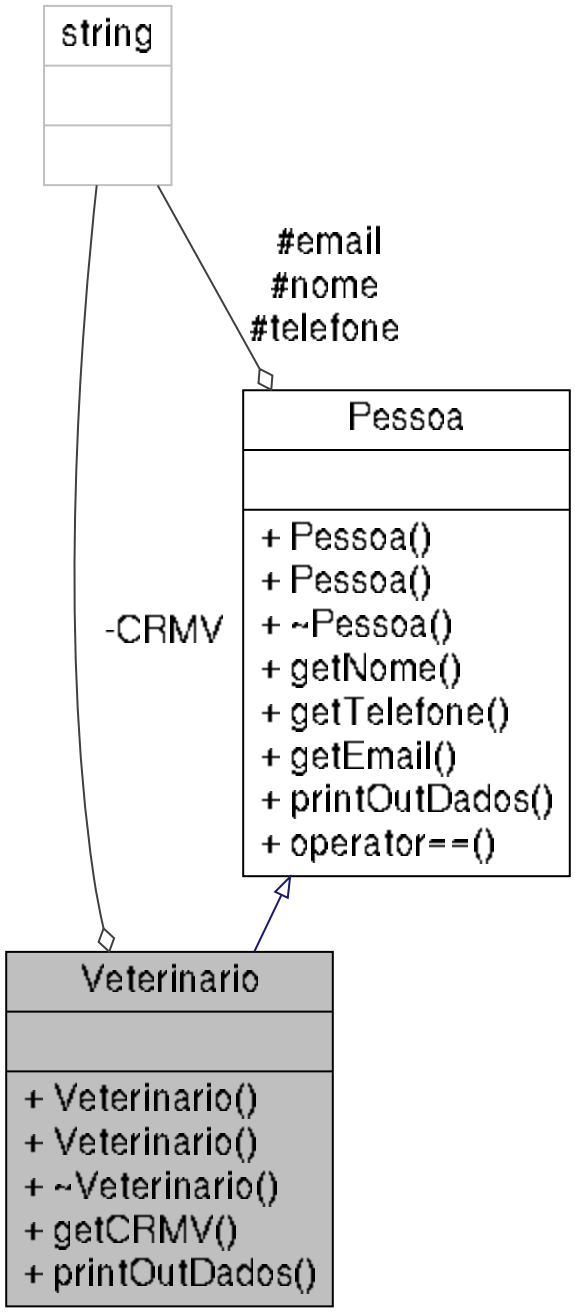


Diagrama de colaboração para Veterinario:



Métodos Públicos

- **Veterinario** (string nome, string telefone, string email, string CRMV)
- string **getCRMV** () const
- ostream & printOutDados (ostream &o) const

  Método virtual para passagem ao Cout, implementado nos herdeiros.

Atributos Privados

- string **CRMV**

Outros membros herdados

4.28.1 Descrição Detalhada

Implementação dos veterinarios.

A implementação para cadastro e admnistração envolve veterinarios credenciados e certificados pelo CRMV. Do mesmo, herdam Pessoa com qual divide atributos básicos comum a Tratador.

4.28.2 Métodos

4.28.2.1 printOutDados()

```
ostream& Veterinario::printOutDados (
            ostream & o ) const  [virtual]
```

Método virtual para passagem ao Cout, implementado nos herdeiros.

Parâmetros

cout	dado pela sobrecarga na classe base.

Retorna

cout usado para impressão em stream.

Implementa Pessoa.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/funcionarios/veterinario.hpp

Índice Remissivo