

Nomes:

Bruno Tavares

Felipe Farias

Guilherme Dias

Natália Georgetti

Quézia Quirino

Exercício 1:

Sistema	
Caso de Uso	CSU01 - Manter Cliente
Atores	Secretária
Descrição	O usuário é cadastrado no sistema.
Pré-condições	A secretária deve estar logada no sistema.
Pós-condições	Usuário é cadastrado.
Fluxo Básico	
1:	Ao clicar em "novo cliente", a secretária digita o CPF do cliente, caso o mesmo seja válido o
2:	sistema exibe uma tela para a mesma inserir nome, endereço e telefone, posteriormente é
3:	adicionar um animal.
Fluxo Alternativo	
1:	Cliente tenta se cadastrar sem documento.
2:	CPF já cadastrado no sistema.
Fluxo de Exceção	
8:	CPF inválido.
9:	Cliente tenta se cadastrar sem ter o animal presente.

Exercício 2:

Sistema	
Caso de Uso	CSU02 - Manter Animal
Atores	Secretária
Descrição	O animal será cadastrado para poder passar em consulta.
Pré-condições	O dono estar cadastrado no sistema.
Pós-condições	atualizar o banco de dados.
Fluxo Básico	
1:	Caso seja um animal doméstico, ao clicar em "adicionar novo animal", o sistema irá solicitar que a
2:	secretária digite o RA, nome, data de nascimento, sexo, espécie, raça, peso e o nome do dono.
Fluxo Alternativo	
1:	O animal chega acompanhado de uma pessoa diferente do dono cadastrado.
2:	Animal já está cadastrado no sistema.
Fluxo de Exceção	
8:	Dono não cadastrado.
9:	Espécie Inválida.

Exercício 3:

Sistema	
Caso de Uso	CSU03 - Manter Veterinário
Atores	Secretária
Descrição O veterinário será cadastrado.	
Pré-condições O veterinário deve possuir CRMV.	
Pós-condições Atualizar no banco de dados.	
Fluxo Básico	
1: Ao clicar em "adicionar novo veterinário" o sistema irá solicitar o CRMV do mesmo, caso ele seja	
2: válido irá solicitar nome, data de nascimento, CPF, telefone e especialidade (caso tiver), após isso	
3: a secretária poderá salvar os dados.	
Fluxo Alternativo	
1: CRMV provisório.	
2: Veterinário com CMV diferente do regional.	
Fluxo de Exceção	
8: CRMV Inválido.	
9: Sociedade emitente do título de concessão de especialidade inexistente.	

Exercício 4:

Sistema	
Caso de Uso	CSU04 - Manter Consulta
Atores	Secretária
Descrição A consulta será agendada.	
Pré-condições Ter um horário/data disponível, o animal e seu dono estarem cadastrados no sistema.	
Pós-condições Atualização do prontuário do animal.	
Fluxo Básico	
1: O sistema irá solicitar dia, horário e procedimento a secretária, caso os mesmos sejam válidos	
2: a secretária irá digitar o tipo do atendimento e os sintomas do animal, ao salvar a consulta estará	
3: marcada.	
Fluxo Alternativo	
1: Uma consulta pode ser adiantada a hora marcada, caso o paciente não compareça.	
2: Caso veterinário falte, reagendar a consulta.	
Fluxo de Exceção	
8: Data inválida.	
9: Dono inválido.	
10: Animal inválido.	

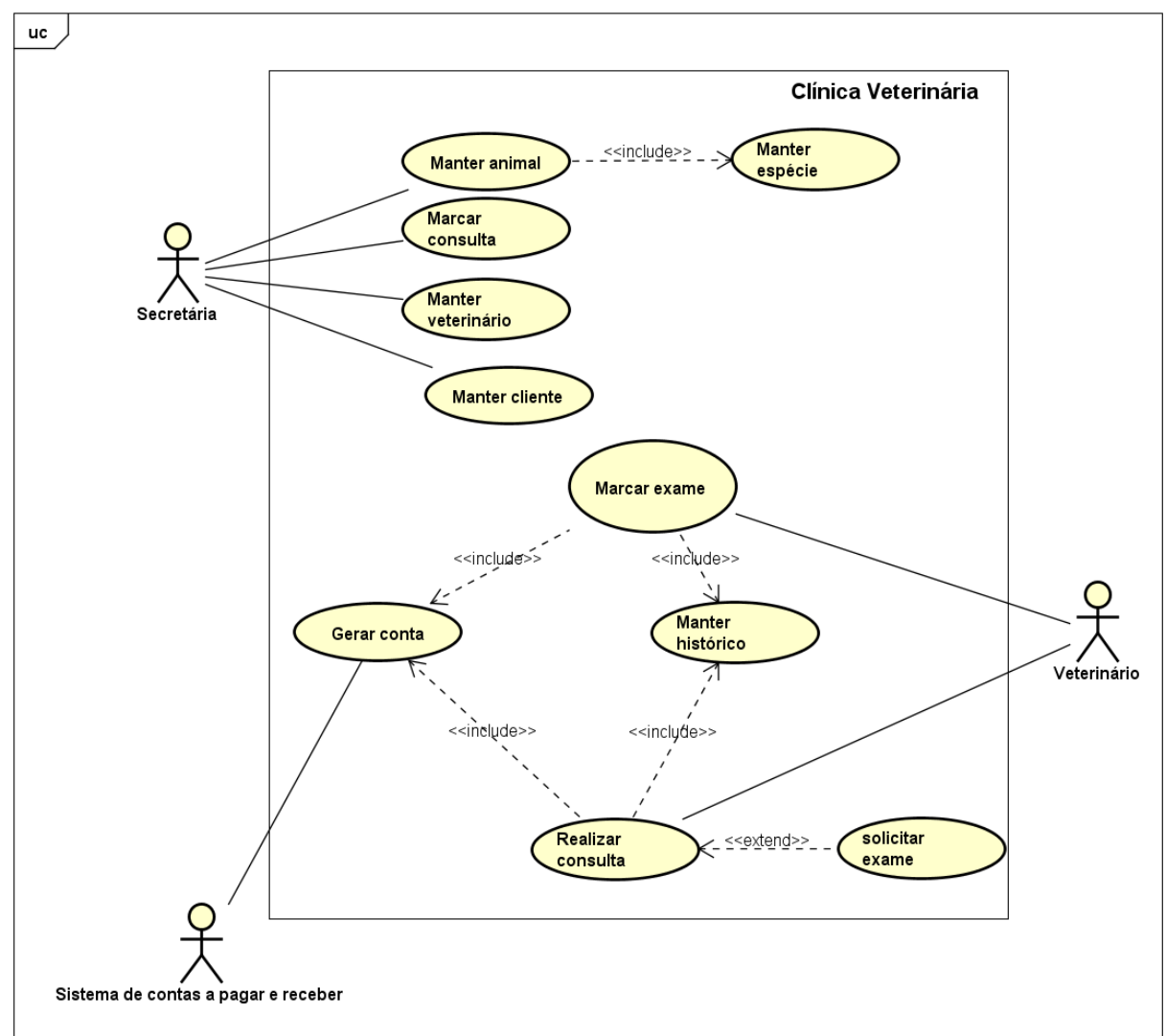
Exercício 5:

Sistema
Caso de Uso CSU05 - Realizar Consulta
Atores Veterinário
Descrição A consulta é realizada quando veterinária, paciente e dono estão presentes
Pré-condições Animal, dono e veterinário estarem cadastrados no sistema.
Pós-condições Atualizar o prontuário do animal.
Fluxo Básico
1: O veterinário irá digitar o procedimento que foi realizado e solicitar exame caso necessário, após
2: isso o sistema irá calcular o valor para que seja confirmado.
Fluxo Alternativo
1: A consulta se estende além do horário estipulado.
2: Consulta reagendada.
3: É solicitado um exame e marcada uma nova consulta após o mesmo.
Fluxo de Exceção
8: O animal chegar ao consultório com outro dono.
9: Método de pagamento inválido.

Exercício 6:

Sistema
Caso de Uso CSU06 – Marcar Exame
Atores Veterinário
Descrição O exame é marcado pelo veterinário e seu resultado será registrado no histórico do animal.
Pré-condições Já ter passado por uma consulta
Pós-condições Atualização do prontuário do animal e agendamento de uma nova consulta.
Fluxo Básico
1: O veterinário irá selecionar os exames que ele necessita e escolher um dia de exame e um dia
2: para o cliente buscar o resultado, caso os mesmos sejam válidos o sistema irá exibir os valores
3: individuais e total dos exames, para que sejam confirmados antes de salvar.
Fluxo Alternativo
1: O veterinário solicita um exame, porém devido a mudanças nos sintomas o mesmo precisa
2: ser remarcado.
Fluxo de Exceção
8: Marcar exame em data/hora inválida.
9: Cliente tenta marcar um exame antes de uma consulta.

Exercício 7:



Exercício 8:

Manter Cliente

Nome

CPF

Endereço

Telefone

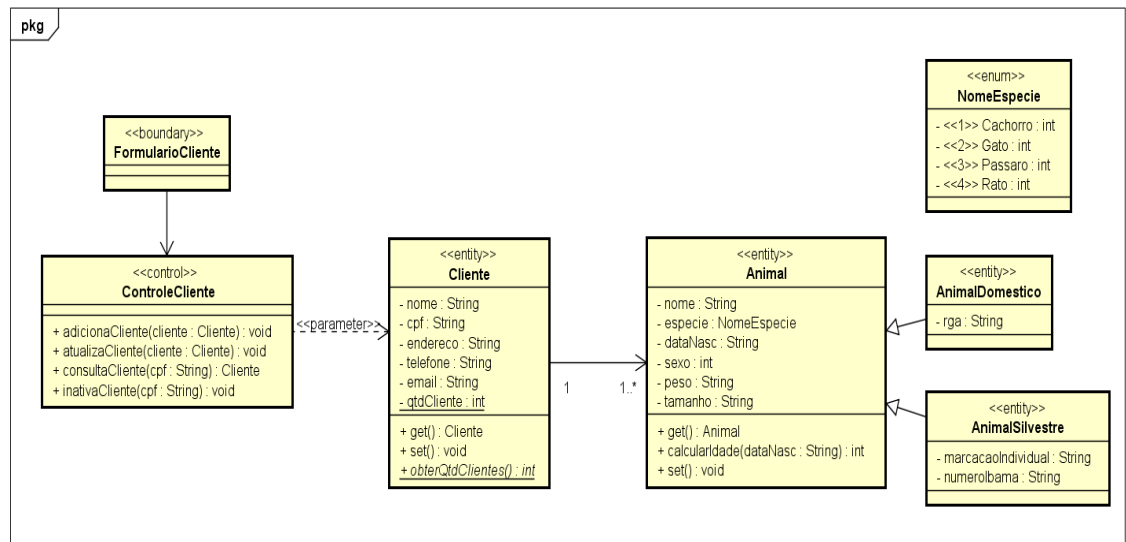
E-mail

Q

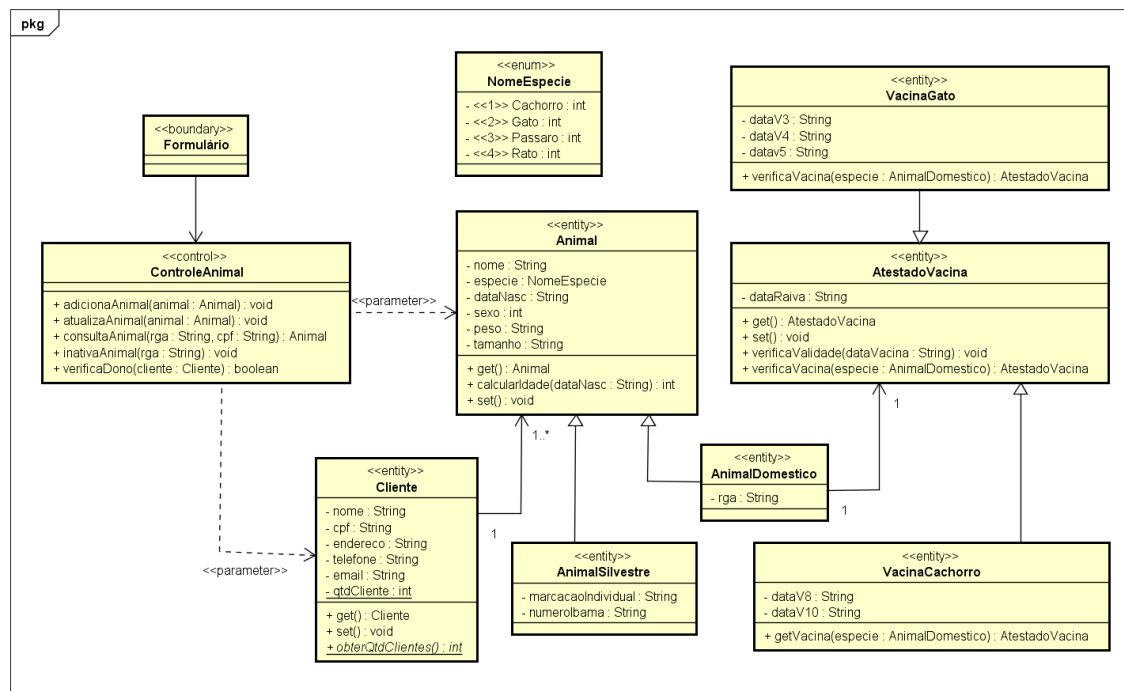
Verificar

Salvar

Cancelar



Exercício 9:



Manter Animal

Tipo Animal:

Domestico

Silvestre

Dono

RGA/Marcação

CPF

Nome

Nome

Data de Nascimento

/

/

Idade:

Sexo

Macho

Peso

kg

Tamanho:

cm

Especie

Canina

Felina

Raça

Labrador

Vacinas

RÁBICA

V3

V4

V5

V8

V10

/

/

/

/

/

/

/

/

/

/

/

/

Em dia?

Em dia?

Em dia?

Em dia?

Em dia?

Em dia?

✓

✓

✓

✗

✗

✓

Salvar

Cancelar

Exercício 10:

Manter Veterinário

CRMV

Nome

Data de Nascimento

/ /

CPF

Q

Telefone

Especialidade

Anestesiologia

Cardiologia

Clinica Geral

Cirurgia geral

Dermatologia

Órgão Expedidor

SBCV

ABROVET

Anestesiologi

Salário: R\$

Disponibilidade

☐ Domingo

☐ Segunda

☐ Terça

☐ Quarta

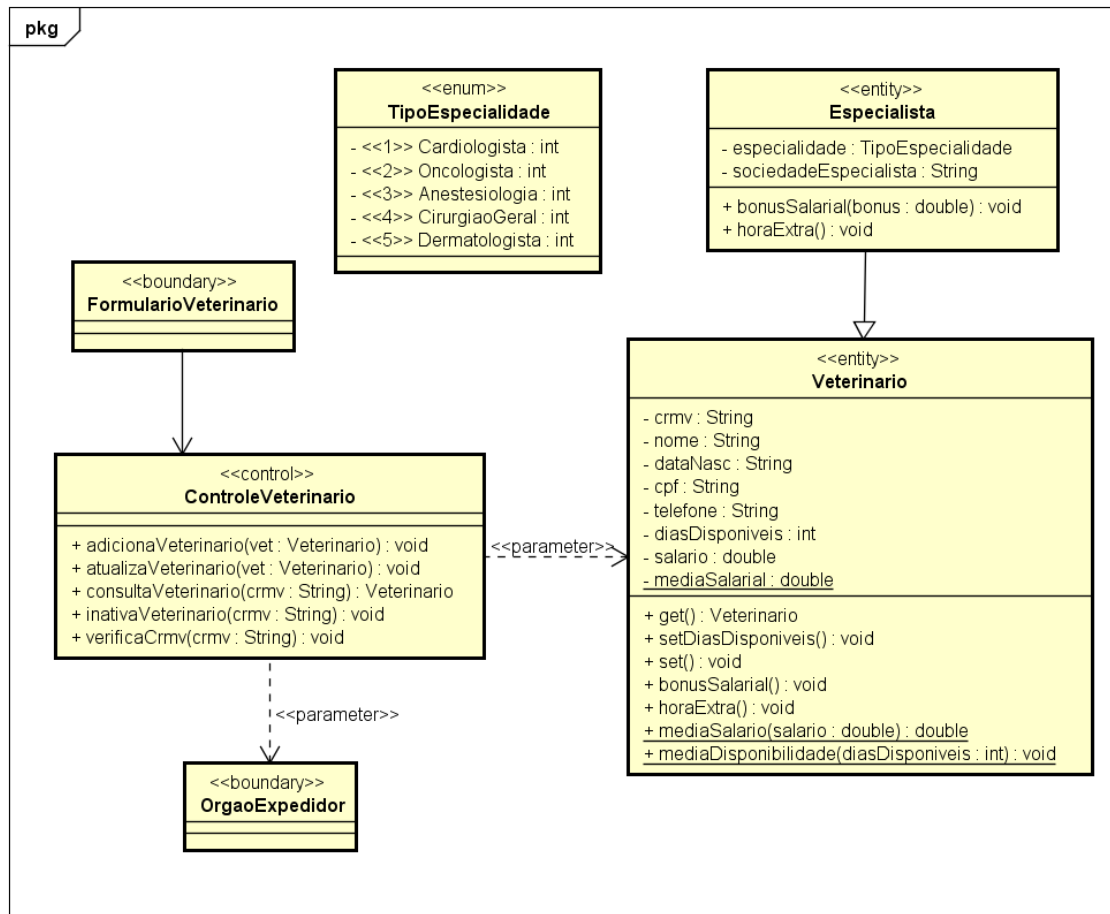
☐ Quinta

☐ Sexta

☐ Sábado

Salvar

Cancelar



Exercício 11:

Marcar Consulta

Dono

CPF

Nome

Selecione Animal

Animal 1

Animal 2

Veterinário

CRMV

Nome

Data

/

/

Hora:

Tipo Atendimento

Primeira Vez

Retorno

Procedimento

Consulta em atenção Básica

Cirurgia de Castração

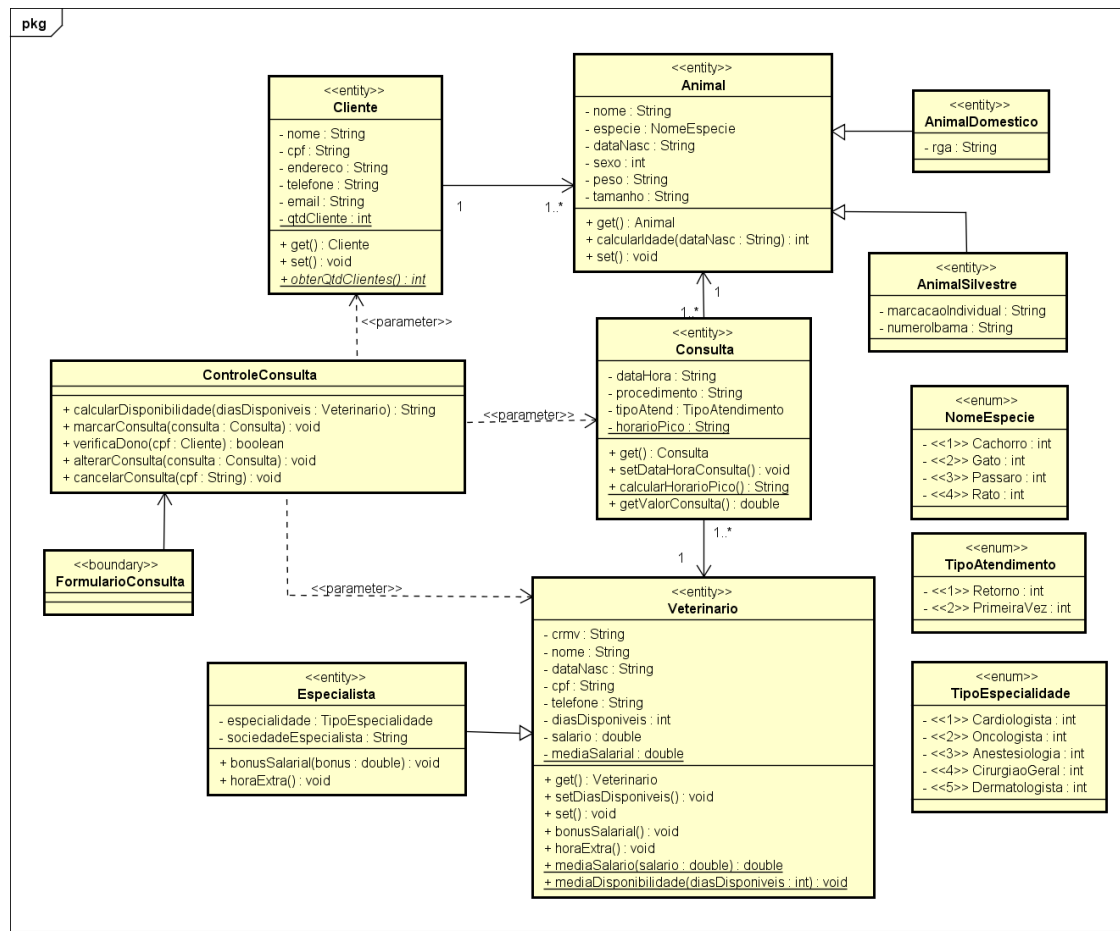
Microcirurgia

Preço Estimado

R\$

Salvar

Cancelar



Exercício 12:

Realizar Consulta

Hora início

:

Prontuário

Hora término

:

Nome animal

Q

RGV/Marcação

CRMV

Sintomas

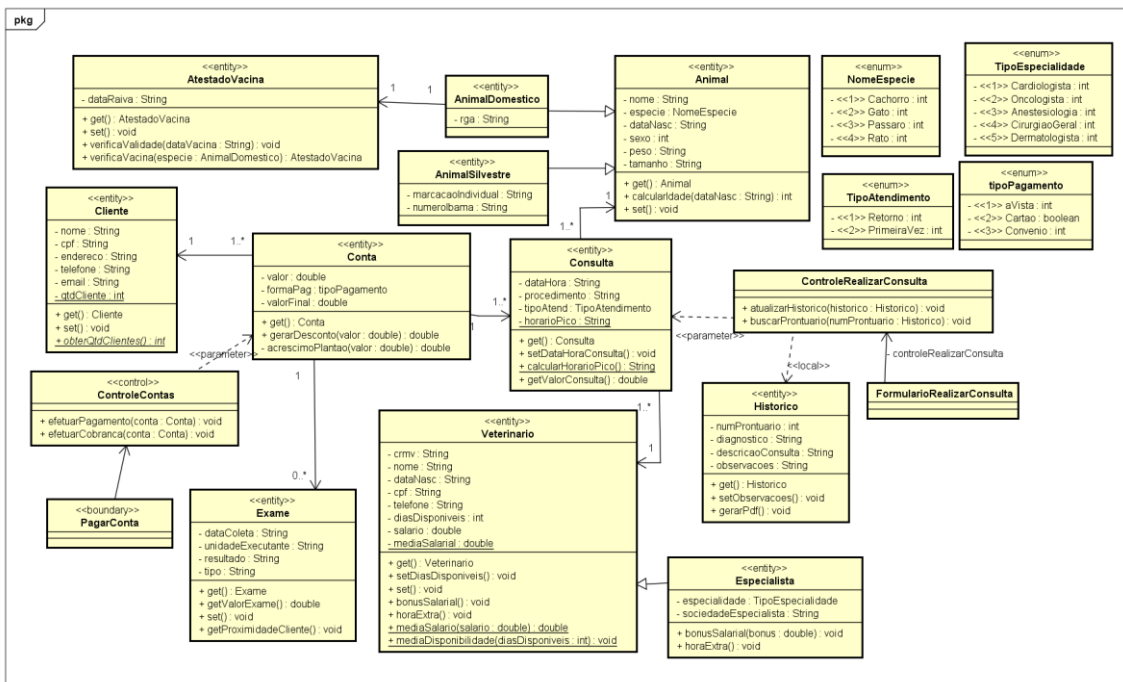
Diagnóstico

Observações


Solicitar Exame

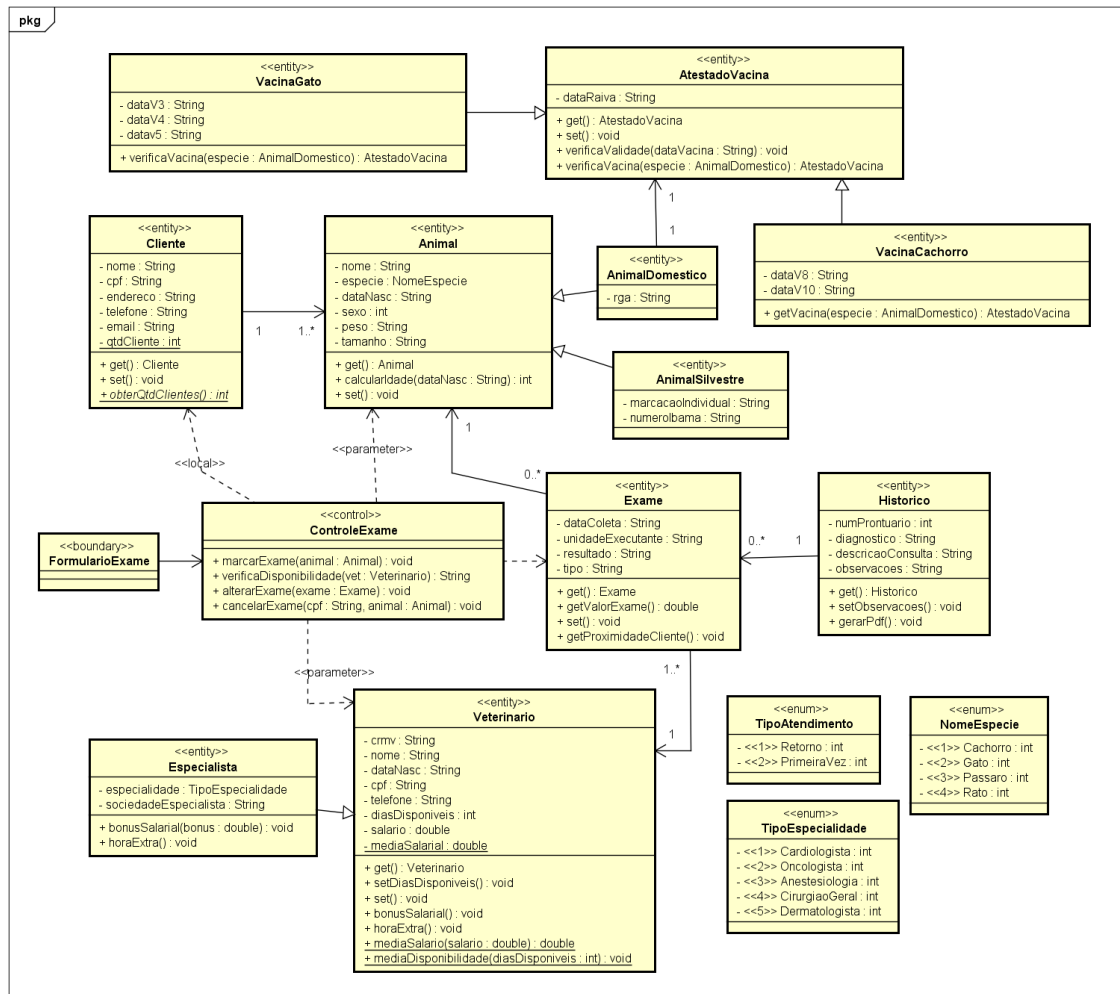
Gravar e Atender

Cancelar

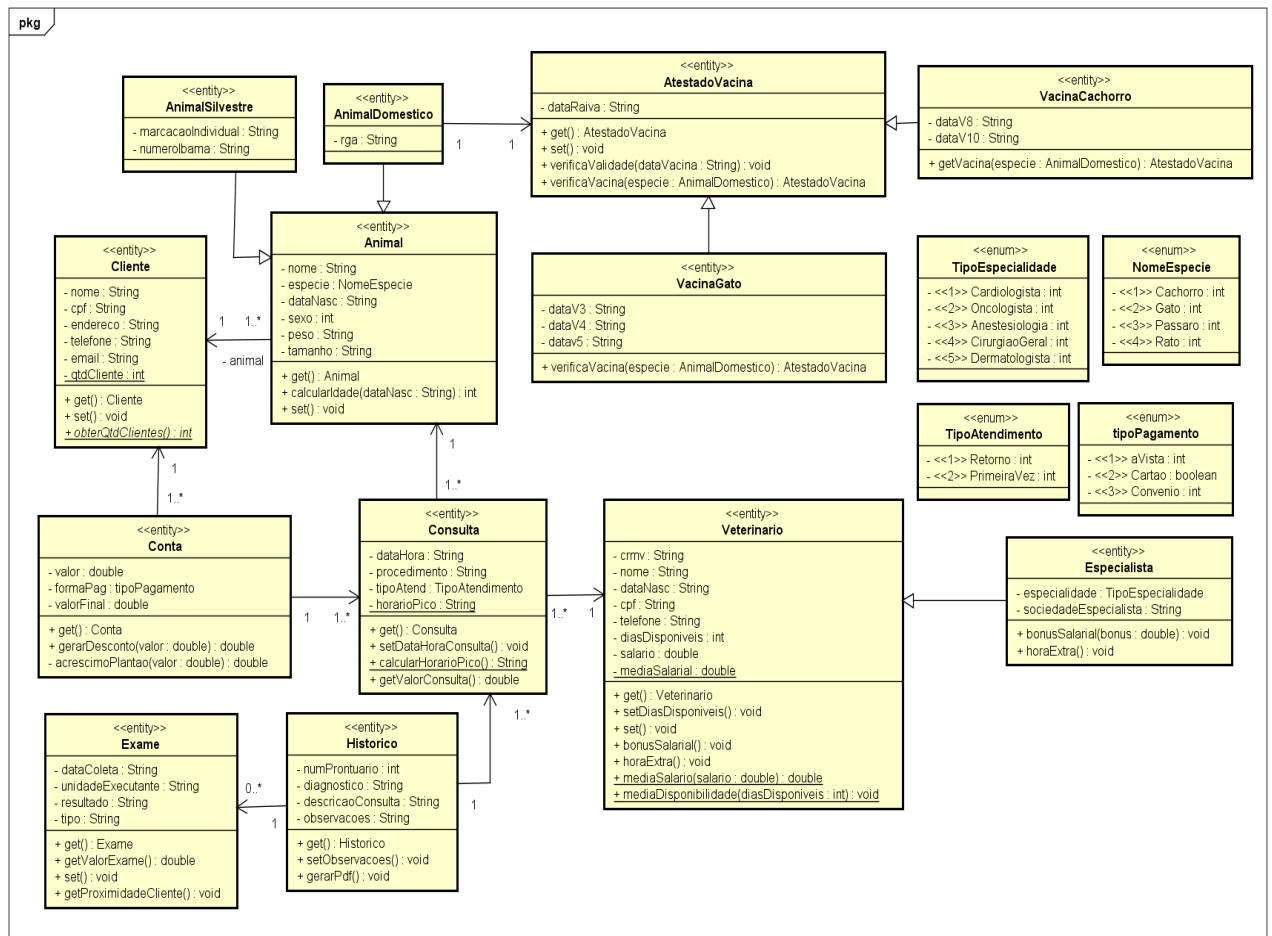


Exercício 13:

Marcar Exame	
Tipo Exame	Data Coleta <input type="text" value="/ /"/> 
<input type="checkbox"/> Ultrassonografia	
<input checked="" type="checkbox"/> Exame De Sangue De Rotina	
<input type="checkbox"/> Exames De Vista	
<input checked="" type="checkbox"/> Exames De Fezes e Urina	Total R\$ <input type="text"/>
Unidade Executante	
<input type="text"/>	
<input type="button" value="Confirmar"/> <input type="button" value="Cancelar"/>	



Exercício 14:



Exercício 15 – Mais coesa é consulta, pois está amarrada em Animal e Veterinario, agrupamento ocorre só porque elas realmente precisam estar juntas para contribuir com algo definido, realizando assim em conjunto as tarefas necessárias. Menos coesa é Historico, pois ela é um histórico de consulta e exame, então ela tem conhecimento de atributos das classes Consulta e Exame.

Exercício 16 – A classe mais acoplada é Consulta, pois depende de Animal e Veterinario para a realização de suas responsabilidades. A classe menos acoplada é Cliente, pois não depende outras classes para funcionar.

Exercício 17 – O método polimórfico bonusSalarial existe, pois há um aumento no bônus quando o veterinário tem uma especialidade. O método verificaVacina para saber qual atestado buscar através do atributo espécie da classe AnimalDomestico. Não violam, pois, a filha pode se passar por mãe.

```
Exercício 18 – public class Especialista extends Veterinario {  
    public double bonusSalarial(double salario){  
        ...  
    }  
}  
Public class VacinaGato extends AtestadoVacina {  
    public boolean verificaVacina (AnimalDomestico especie) {  
        ...  
    }  
}
```

Exercício 19 – A classe TipoEspecialidade foi feita para evitar erros na hora de inserção no banco e para limitar à apenas as especialidades que o domínio atende;

A classe TipoAtendimento limita a somente dois tipos de atendimento, que são Primeira Vez e Retorno;

A classe TipoEspecie limita a seleção somente aos tipos de animais que são atendidos no domínio;

A classe TipoPagamento aceita apenas as formas de pagamento aceitas pelo domínio;

```
Exercício 20 – public enum TipoEspecialidade {  
    Cardiologista, Oftalmologista, Anestesiologia, CirurgiaoGeral,  
    Dermatologia;  
}  
    public enum TipoAtendimento {  
        Retorno, PrimeiraVez;  
    }  
    public enum TipoEspecie {  
        Cachorro, Gato, Passaro, Roedor, }
```

```
public enum TipoPagamento {
    aVista, Cartao, Convenio;
}
```

Exercício 21 – O atributo `mediaSalarial` na classe `Veterinario`, guarda uma média de salário entre todos os objetos do tipo `Veterinario`;
 O atributo `horarioPico` na classe `Consulta` foi feito, pensando em qual horário a clínica atende mais pacientes, podendo até mesmo remanejar o pessoal de forma a atender a demanda.
 O atributo `QtdClientes` na classe `Cliente`, guarda o valor do total de cliente que a clínica tem em seu banco de dados.
 O Método `mediaSalarial` calcula a média salarial de todos os veterinários.
 O método `mediaDisponibilidade` calcula qual a média de disponibilidade de todos os veterinários, para saber qual dia tem mais veterinário disponível.
 O método `obterQtdClientes` na classe `Cliente` calcula a quantidade de clientes que possui no banco de dados.

Exercício 22 – `private static double mediaSalarial;`
`private static int quantidadeClientes;`
`private static String horarioPico;`

```
public static double mediaSalarial() {
    ...
}
public static String calcularHorarioPico() {
    ...
}
public static int obterQtdClientes() {
    ...
}
```

Exercício 23 – A vantagem dessa dependência é aumento no desempenho, a desvantagem é queda no encapsulamento e aumento no acoplamento.

Exercício 24 –

```

public class Animal {
    private Cliente cliente;
    ...
}
public class Consulta {
    private Animal animal;
    private Veterinario veterinario;
}
public class Historico {
    private Consulta consulta;
    private Exame exame;
}
public class Conta {
    private Consulta consulta;
    private Exame exame;
    private Cliente cliente;
}
public class AnimalDomestico extends Animal {
    private AtestadoVacina atestadoV;
}

```

Exercício 25 – A vantagem é aumento no encapsulamento e queda no acoplamento, a desvantagem é o baixo desempenho.

Exercício 26 –

```

public class ControleCliente {
    public void adicionaCliente(Cliente c) {
        ...
    }

    public void consultaCliente(Cliente c) {
        ...
    }
    public void atualizaCliente(Cliente c) {
        ...
    }
    public void inativaCliente(Cliente c) {
        ...
    }
}
public class ControleConta {
    public void efetuarPagamento(Conta conta) {
        ...
    }
    public void efetuarCobranca(Conta conta) {
        ...
    }
}

```

```
public class ControleAnimal {  
    public void adicionaAnimal(Animal a) {  
        ...  
    }  
    public void consultaAnimal(Animal a) {  
        ...  
    }  
    public void atualizaAnimal(Animal a) {  
        ...  
    }  
    public void inativaAnimal(Animal a) {  
        ...  
    }  
    public boolean verificaDono(Cliente c) {  
        ...  
    }  
}  
  
public class ControleVeterinario {  
    public void adicionaVeterinario(Veterinario v) {  
        ...  
    }  
    public void consultaVeterinario(Veterinario v) {  
        ...  
    }  
    public void atualizaVeterinario(Veterinario v) {  
        ...  
    }  
    public void inativaVeterinario(Veterinario v) {  
        ...  
    }  
    public void verificaCrmv(String crmv) { ... } }
```

```

public class ControleConsulta {
    public String calculaDisponibilidade(Veterinario v) {
        ...
    }
    public void marcarConsulta (Consulta c) {
        ...
    }
    public boolean verificaDono(Cliente c) {
        ...
    }
    public void alterarConsulta(Consulta c) {
        ...
    }
    public void cancelarConsulta(String cpf) {
        ...
    }
}

public class ControleExame {
    public void marcarExame(Animal a) {
        ...
    }
    public String verificaDisponibilidade(Veterinario v) {
        ...
    }
    public void alterarExame(Exame e) {
        ...
    }
    public void cancelarExame(String cpf, Animal a) {
        ...
    }
}

```

```

public class ControleRealizarConsulta {
    public void buscarProntuario(Historico h) {
        ...
    }
}

```

Exercício 27 – A vantagem de guardar a instância do objeto em variável local vem quando se precisa acessar o mesmo objeto muitas vezes, assim evitando de ficar criando o objeto toda vez que for manipulá-lo.

```

Exercício 28 – public class ControleCliente {
    public void adicionaCliente() {
        Cliente c = new Cliente();
        ...
    }

    public void consultaCliente() {
        Cliente c = new Cliente();
        ...
    }

    public void atualizaCliente() {
        Cliente c = new Cliente();
        ...
    }

    public void inativaCliente() {
        Cliente c = new Cliente();
        ...
    }
}

public class ControleConta {
    public void efetuarPagamento() {
        Conta c = new Conta();
        ...
    }

    public void efetuarCobranca() {
        Conta c = new Conta();
        ...
    }
}

```

```
public class ControleAnimal {  
    public void adicionaAnimal() {  
        Animal a = new Animal();  
        ...  
    }  
    public void consultaAnimal() {  
        Animal a = new Animal();  
        ...  
    }  
    public void atualizaAnimal() {  
        Animal a = new Animal();  
        ...  
    }  
    public void inativaAnimal() {  
        Animal a = new Animal();  
        ...  
    }  
    public boolean verificaDono() {  
        Cliente c = new Cliente();  
        ...  
    }  
}  
  
public class ControleVeterinario {  
    public void adicionaVeterinario() {  
        Veterinario v = new Veterinario();  
        ...  
    }  
    public void consultaVeterinario() {  
        Veterinario v = new Veterinario();  
        ...  
    }  
}
```

```

    public void atualizaVeterinario() {
        Veterinario v = new Veterinario();
        ...
    }
    public void inativaVeterinario() {
        Veterinario v = new Veterinario();
        ...
    }
}

public class ControleConsulta {
    public String calculaDisponibilidade() {
        Veterinario v = new Veterinario();
        ...
    }
    public void marcarConsulta () {
        Consulta c = new Consulta();
        ...
    }
    public boolean verificaDono() {
        Cliente c = new Cliente();
        ...
    }
    public void alterarConsulta() {
        Consulta c = new Consulta();
        ...
    }
}

```

```

public class ControleExame {
    public void marcarExame() {
        Animal a = new Animal();
        ...
    }
    public String verificaDisponibilidade() {
        Veterinario v = new Veterinario();
        ...
    }
    public void alterarExame() {
        Exame e = new Exame();
        ...
    }
    public void cancelarExame() {
        Animal a = new Animal();
        ...
    }
}

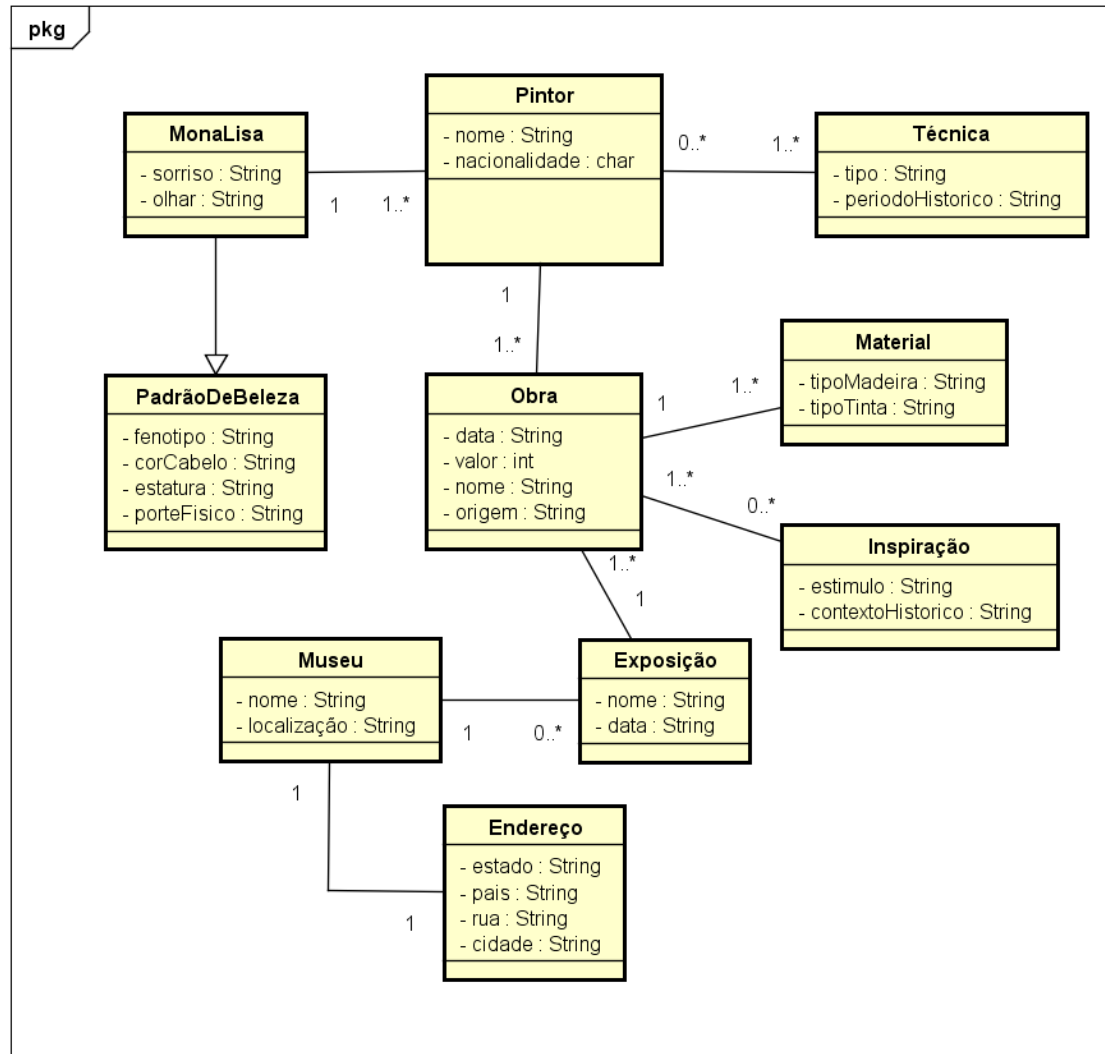
public class ControleRealizarConsulta {
    public void buscarProntuario(Historico h) {
        Historico h = new Historico();
        ...
    }
}

public class ControleRealizarConsulta {
    public void atualizarHistorico() {
        Historico h = new Historico();
        ...
    }
}

```

Parte B:

Exercício 1:



Exercício 2:

