Engenharia de Software III

Bruno Bega Harnik	1110481823052
Fernanda Pinheiro Reis	1110481823022
Luiz Fernando Geraldo dos Santos	1110481823051
Raquel Martins do Nascimento	1110481823032

São Paulo
2020

Lista de Exercícios para a N1

1- Especifique textualmente o caso de uso Manter Cliente (CSU01), apresentando os fluxos (cenários) principal, alternativo e de exceção, de acordo com o template disponibilizado.

Sistema	Pet Shop
Caso de Uso	CSU 01 – Manter Cliente
Atores	Secretario, Cliente, Animal
Requisitos	

Descrição

Um cliente chega à clinica veterinária com seu pet e precisa informar seus dados para registro no sistema.

Pré-condições

Pós-condições

Cliente deve passar as informações de seu pet para registro no sistema da clínica veterinária.

	Fluxo Básico
1	Cliente informa dados para secretário.
2	Secretário verifica se cliente já está cadastrado no sistema. Se não, segue o fluxo principal. Se sim, vai para o alternativo.
3	Secretário cadastra nome, cpf, rg, data de nascimento, endereço e telefone do cliente
4	
5	
6	
7	
8	
9	

	Fluxo Alternativo
1	Cliente já está cadastrado no sistema.
2	Verifica se dados estão atualizados. Se sim, segue para ManterPet().
3	Se não, secretário altera dados inconsistentes do cliente.
4	
5	
6	
7	

	Fluxo de Exceção
8	CPF inválido.
9	Cadastro não concluído.
10	
11	
12	
28	

Sistema	Pet Shop
Caso de Uso	CSU 01.01 – Validar CPF
Atores	Secretario, Cliente
Requisitos	

Descrição

Secretário realiza consulta automática integrada ao sistema no site da receita federal, para verificar se o cpf é válido.

Pré-condições

ManterCliente() deve ter sido iniciado.

Pós-condições

Cliente está apto ou não a prosseguir com seu cadastro na clínica.

	Fluxo Básico
1	Cliente informa cpf e data de nascimento para secretário.
2	Secretário verifica se o CPF é válido por validação automática do sistema.
3	Se sim, retorna ao Manter Cliente()
4	Se não, vai ao fluxo alternativo.
5	
6	
7	
8	
9	

	Fluxo Alternativo
1	Caso o CPF não seja válido, o Secretário faz a escolha se quer ou não prosseguir com o cadastro do cliente;
2	
3	
4	
5	
6	
7	

	Fluxo de Exceção
8	Secretário decidiu que não cadatstrará o cliente.
9	
10	
11	
12	
28	

2- Especifique textualmente o caso de uso Manter Animal (CSU02), apresentando os fluxos (cenários) principal, alternativo e de exceção, de acordo com o template disponibilizado. O caso de uso Manter Espécie deve ser adequado e especificado como caso de uso incluído (<<include>>) do caso de uso principal, isto é, o CSU02.

Sistema	Pet Shop
Caso de Uso	CSU 02 – Manter Animal
Atores	Secretário, Cliente, Animal
Requisitos	Manter Cliente
Descrição	
Cliente, após informar seus dados para cadastro, começa o cadastro de seu pet.	
Pré-condições	
ManterCliente já deve estar concluído.	
Pós-condições	
Fluxo Básico	
1	Cliente informa o nome do animal para secretário verificar se o animal já está cadastrado no sistema.
2	Se sim, vai ao fluxo alternativo.
3	Se não, cliente informa os dados nome, idade, peso, raça e espécie para o secretário cadastrar no sistema.
4	Secretário consulta a lista de espécies que são atendidas por aquela equipe veterinária.
5	Vai ao caso de uso Consultar Espécie.
6	Caso tudo ocorra conforme o planejado, secretário conclui o cadastro do animal.
7	
Fluxo Alternativo	
1	Secretário verifica se os dados do animal estão atualizados. Se sim, segue ao próximo caso de uso.
2	Se não, atualiza com os dados informados pelo cliente.
3	
32	
Fluxo de Exceção	
8	Espécie não pode ser atendida pelo corpo de profissionais disponível.
9	

Sistema	Pet Shop
Caso de Uso	CSU 02.01 – Consultar Especie (ManterEspecie)
Atores	Secretário, Cliente
Requisitos	Deve ter iniciado o cadastro do animal (ManterAnimal())
Descrição	
Secretário verifica se aquela espécie de animal é atendida pelo corpo de veterinários disponível.	
Pré-condições	
Espécies anteriormente cadastradas	
Pós-condições	
Prosseguir no caso de uso Manter Animal().	
Fluxo Básico	
1	Secretário verifica se a espécie informada está cadastrada no sistema. Se sim, seleciona e retorna ao cadastro do
2	Se não, verifica se o corpo de veterinários possui a especialidade para lidar com aquele tipo de animal
3	Se sim, seleciona e retorna ao cadastro do animal. Se não, vai ao fluxo de exceção.
4	
5	
6	
7	
8	
9	
10	
29	
30	
Fluxo Alternativo	
1	
2	
3	
4	
5	
6	
7	
8	
9	
32	
Fluxo de Exceção	
8	Especialidade não é atendida pelo corpo de veterinários.
9	
10	
11	
12	
13	
14	
15	
28	

3- Especifique textualmente o caso de uso Manter Veterinário (CSU03), apresentando os fluxos (cenários) principal, alternativo e de exceção, de acordo com o template disponibilizado.

Sistema	Pet Shop
Caso de Uso	CSU 03 – Manter Veterinario
Atores	Veterinário, Secretário
Requisitos	
Descrição	
Secretário realiza o cadastro de um novo veterinário no sistema.	
Pré-condições	
Pós-condições	
Terá acesso a outros casos de uso, como realizar consulta e agendar exame.	
Fluxo Básico	
1	Secretário solicita os dados de cadastro do veterinário.
2	Veterinário fornece nome e CPF ao secretário.
3	Vai ao CSU01.01 para validar o CPF do veterinário.
4	Se o veterinário já estiver cadastrado, vai ao fluxo alternativo.
5	Se não, solicita também data de nascimento, CRMV, endereço, telefone, especialidades.
6	Termina o preenchimento e insere o veterinário na lista de veterinários da clínica, bem como suas especialidades.
7	
8	
9	
10	
11	
Fluxo Alternativo	
1	Verifica se os dados informados pelo veterinário estão corretos. Caso estejam, segue ao próximo caso de uso.
2	Se não, verifica e atualiza os dados inconsistentes.
5	
6	
7	
8	
9	
10	
11	
Fluxo de Exceção	
8	Veterinário não possui CPF.
9	Veterinário não possui CRMV.
10	
11	
12	
13	
14	

Sistema	Pet Shop
Caso de Uso	CSU 01.01 – Validar CRMV
Atores	Secretário, veterinário
Requisitos	

Descrição

Secretário realiza consulta automática integrada ao sistema no site do CRMV estadual, para verificar se o crmv é válido.

Pré-condições

ManterVeterinario() deve ter sido iniciado.

Pós-condições

Veterinário está apto ou não a prosseguir com seu cadastro na clínica.

	Fluxo Básico
1	Veterinário informa crmv para secretário.
2	Secretário verifica se o CRMV é válido por validação automática do sistema.
3	Se sim, retorna ao Manter Veterinario()
4	Se não, vai ao fluxo alternativo.
5	
6	
7	
8	
9	

	Fluxo Alternativo
1	Caso o CRMV não seja válido, o Secretário faz a escolha se quer ou não prosseguir com o cadastro do veterinário.
2	
3	
4	
7	

	Fluxo de Exceção
8	Secretário decidiu que não cadastrará o veterinário.
9	
10	
11	
12	
28	

4- Especifique textualmente o caso de uso Marcar Consulta (CSU04), apresentando os fluxos (cenários) principal, alternativo e de exceção, de acordo com o template disponibilizado.

Sistema	Pet Shop
Caso de Uso	CSU 04 – Marcar Consulta
Atores	Secretario, Animal, Cliente, Veterinario
Requisitos	
Descrição	
Cliente chega à clínica para agendar consulta para seu Pet.	
Pré-condições	
Cliente, animal e veterinário devem estar cadastrados no sistema.	
Pós-condições	
Fluxo Básico	
1	Cliente chega à clínica e informa seu nome. Se estiver cadastrado, trará a lista de pets cadastrados em seu nome.
2	Secretário seleciona o pet para aquela consulta.
3	Secretário verifica se o cliente tem preferência de veterinário.
4	Secretário verifica os veterinários disponíveis para aquele tipo de Pet.
5	Secretário consulta a agenda do Consultório para verificar se há datas disponíveis para aquele Veterinario
6	Secretário confirma consulta com o Cliente.
7	
16	
30	
Fluxo Alternativo	
1	Se o Cliente não estiver cadastrado, chamará o caso "manter Cliente"
2	Se a lista de pets não retornar o pet atrelado ao cliente, deve chamar o caso "manter Animal"
3	
32	
Fluxo de Exceção	
8	Não há datas disponíveis para consulta.
9	
10	
28	

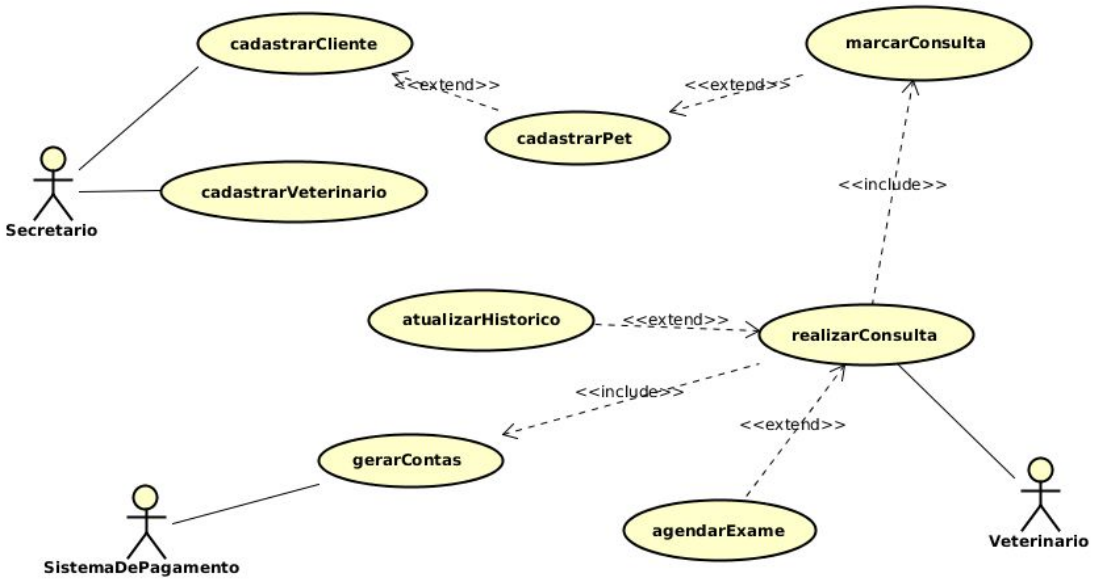
5- Especifique textualmente o caso de uso Realizar Consulta (CSU05), apresentando os fluxos (cenários) principal, alternativo e de exceção, de acordo com o template disponibilizado.

Sistema	Pet Shop
Caso de Uso	CSU 05 – Realizar Consulta
Atores	Secretário, Veterinário
Requisitos	ManterCliente, ManterAnimal, ManterVeterinario, AgendarConsulta
Descrição	
Veterinário realiza consulta agendada para determinado pet.	
Pré-condições	
Pós-condições	
Registros atualizados e consulta realizada	
Fluxo Básico	
1	Cliente informa os sintomas do animal,
2	Secretária registra os sintomas no histórico da consulta do animal,
3	Encaminha para o veterinário,
4	Veterinário confere o histórico da consulta do animal,
5	Veterinário verifica exames (opcional),
6	Veterinário analisa as complicações e necessidades do animal
7	Veterinário solicita Exames necessários (opcional),
8	Atualiza Histórico (com sintomas, exames, diagnostico),
9	Secretária verifica se há exames ou novas consultas para marcar,
10	Finaliza a consulta e gera a conta a pagar.
11	
12	
13	
Fluxo Alternativo	
1	Realiza o exame agendado,
2	Aplica medicação (opcional),
3	Verifica necessidade de novo exame,
4	Se sim: Continua no 07 do Fluxo Principal.
5	Se não: Continua no 08 do Fluxo Principal.
6	
7	
8	
9	
10	
Fluxo de Exceção	
8	Sistema de pagamento instável
9	Falta na consulta devido ausência de um dos integrantes
10	
11	
12	
13	
14	

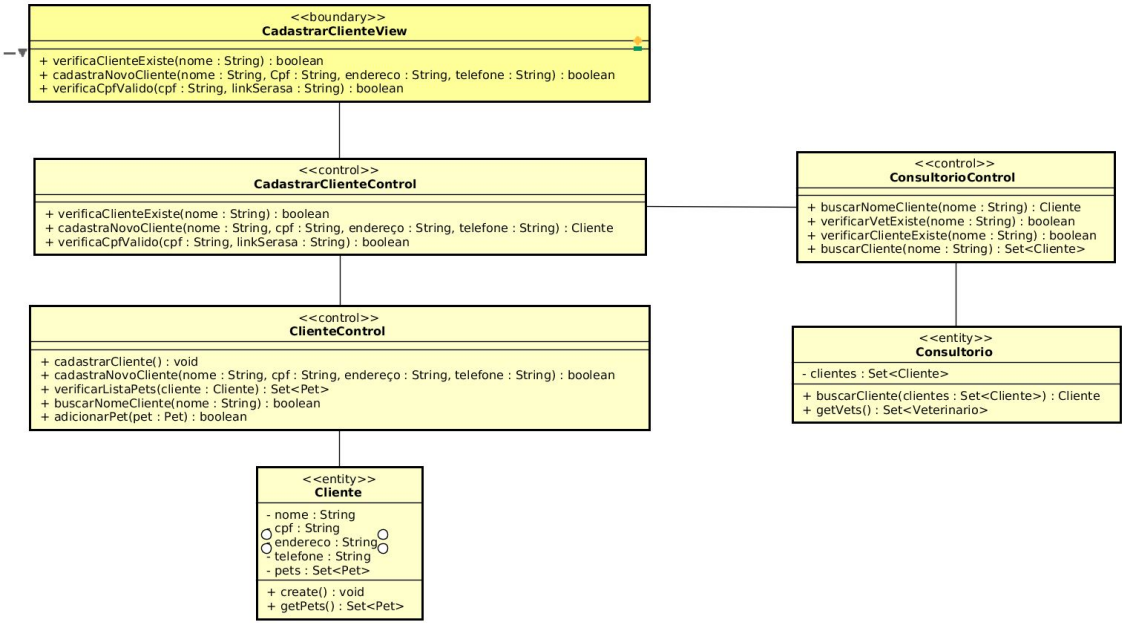
6- Especifique textualmente o caso de uso Marcar Exame (CSU06), apresentando os fluxos (cenários) principal, alternativo e de exceção, de acordo com o template disponibilizado. Este caso de uso deve ser adaptado e especificado como caso de uso principal, retirando o mesmo como caso de uso estendido (<<extend>>) do caso de uso Realizar Consulta (CSU05).

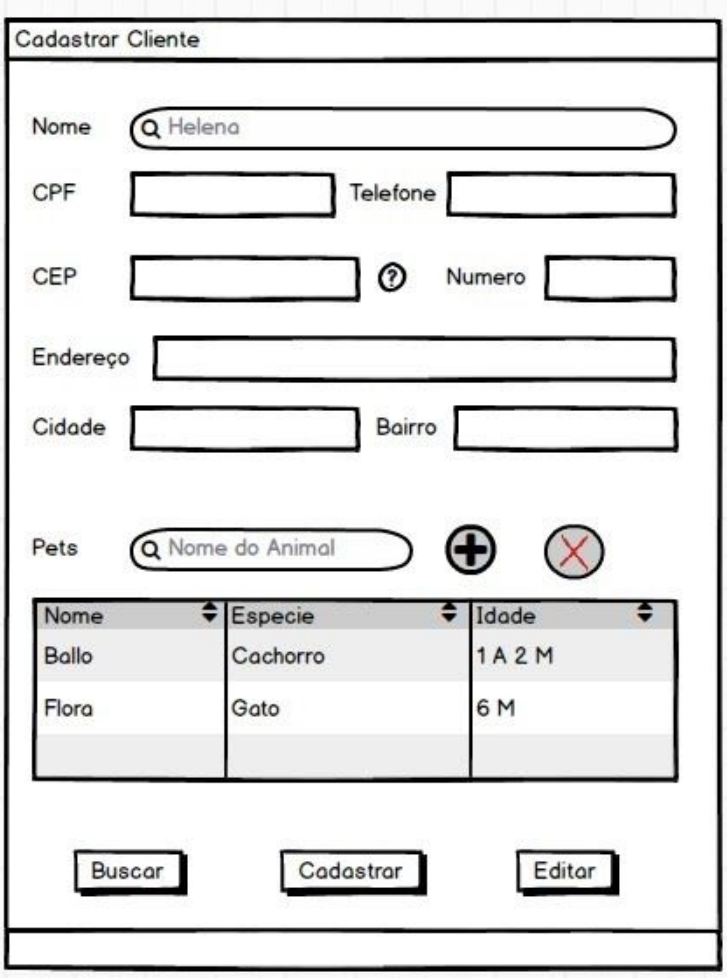
Sistema	Pet Shop
Caso de Uso	CSU 06 – Solicitar exame
Atores	Veterinário
Requisitos	ManterCliente, ManterAnimal, ManterVeterinario, AgendarConsulta
Descrição	
Veterinário solicita exame do animal para mais informações	
Pré-condições	
Cliente e Animal cadastrados no sistema	
Pós-condições	
Exame solicitado e histórico atualizado	
Fluxo Básico	
1	Veterinário solicita exame do animal,
2	Informa nome do exame e detalhes da solicitação,
3	Aplica medicação (opcional),
4	Atualiza Histórico (com sintomas, exames, diagnostico),
5	Secretária verifica se há exames ou novas consultas para marcar,
6	Finaliza a consulta e gera a conta a pagar.
7	
8	
9	
10	
11	
12	
13	
Fluxo Alternativo	
1	Veterinário verifica último exame solicitado
2	Analisa se precisa de outro exame ou aguarda a conclusão do exame pendente
3	Se sim: Informa o nome do exame e detalhes da solicitação,
4	Continua no 04 do Fluxo Principal
5	Se não: Aguarda a conclusão Continua no 04 do Fluxo Principal
6	
7	
8	
9	
10	
Fluxo de Exceção	
8	
9	
10	

7- Modele um novo Diagrama de Casos de Uso (DCU) com base nas especificações textuais.

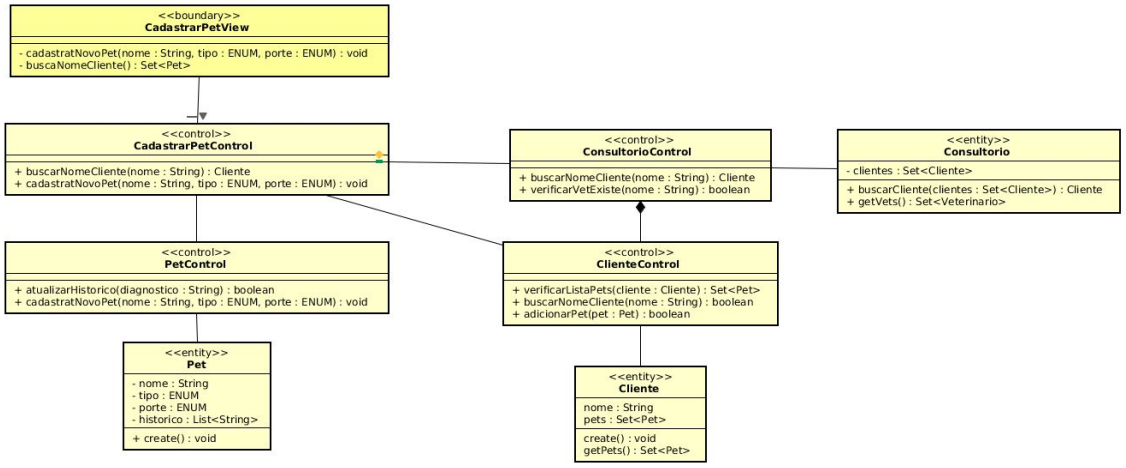


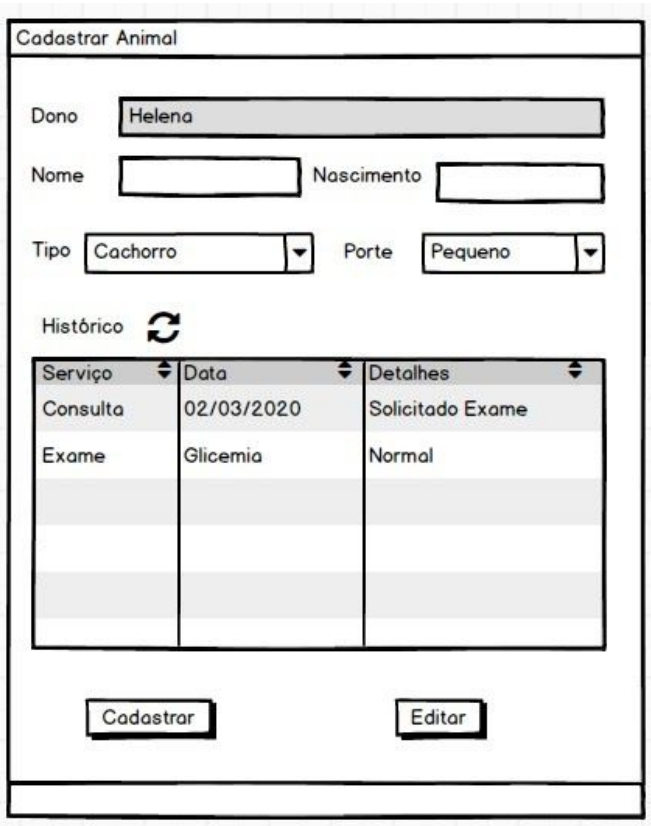
8- Modele uma VCP para o CSU01, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos. Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU01.


<<boundary>>
CadastrarClienteView
+ verificaClienteExiste(nome : String) : boolean
+ cadastraNovoCliente(nome : String, Cpf : String, endereco : String, telefone : String) : boolean
+ verificaCpfValido(cpf : String, linkSerasa : String) : boolean
<<control>>
CadastrarClienteControl
+ verificaClienteExiste(nome : String) : boolean
+ cadastraNovoCliente(nome : String, cpf : String, endereço : String, telefone : String) : Cliente
+ verificaCpfValido(cpf : String, linkSerasa : String) : boolean
<<control>>
ConsultorioControl
+ buscarNomeCliente(nome : String) : Cliente
+ verificarVetExiste(nome : String) : boolean
+ verificarClienteExiste(nome : String) : boolean
+ buscarCliente(nome : String) : Set<Cliente>
<<control>>
ClienteControl
+ cadastrarCliente() : void
+ cadastraNovoCliente(nome : String, cpf : String, endereço : String, telefone : String) : boolean
+ verificarListaPets(cliente : Cliente) : Set<Pet>
+ buscarNomeCliente(nome : String) : boolean
+ adicionarPet(pet : Pet) : boolean
<<entity>>
Consultorio
- clientes : Set<Cliente>
+ buscarCliente(clientes : Set<Cliente>) : Cliente
+ getVets() : Set<Veterinario>
<<entity>>
Cliente
- nome : String
- cpf : String
- endereco : String
- telefone : String
- pets : Set<Pet>
+ create() : void
+ getPets() : Set<Pet>

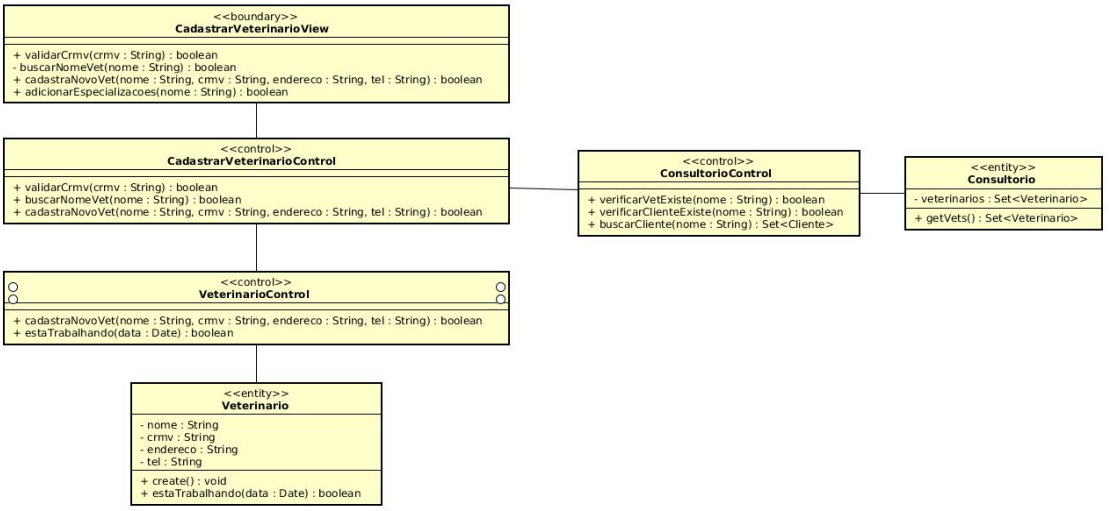

Cadastrar Cliente
Nome
Helena
CPF
Telefone
CEP
Numero
Endereço
Cidade
Bairro
Pets
Nome do Animal
Nome
Especie
Idade
Ballo
Cachorro
1 A 2 M
Flora
Gato
6 M
Buscar
Cadastrar
Editar

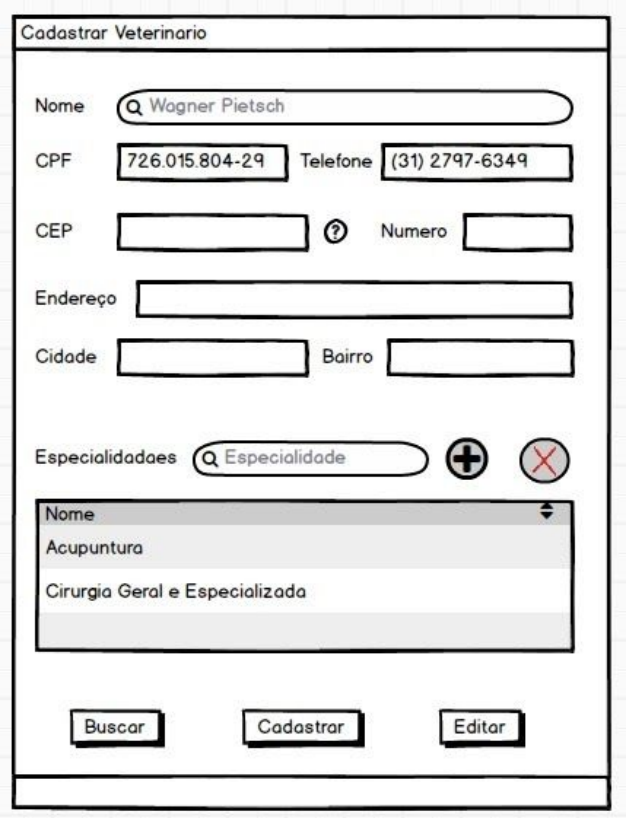
9- Modele uma VCP para o CSU02, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos. Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU02.



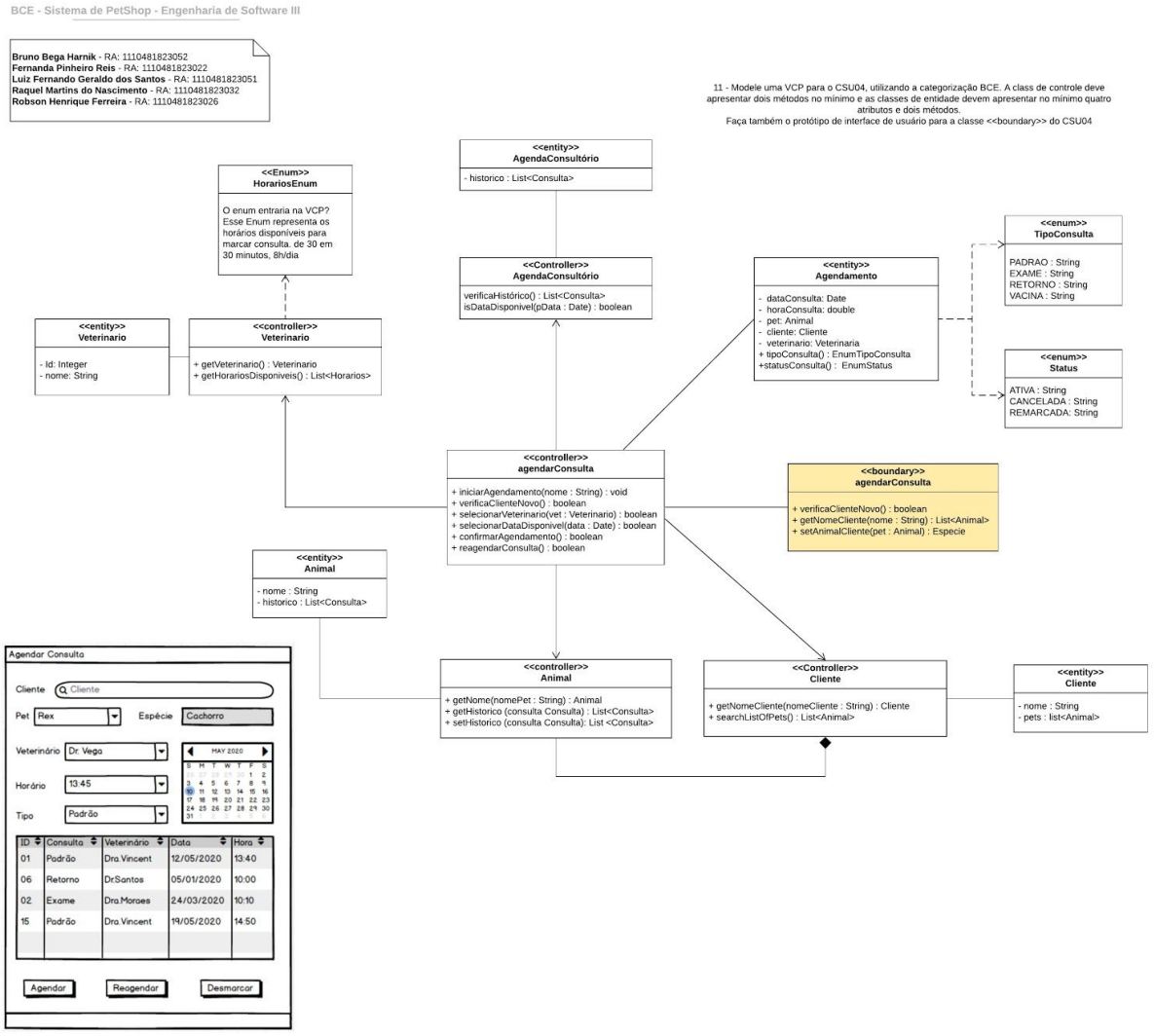


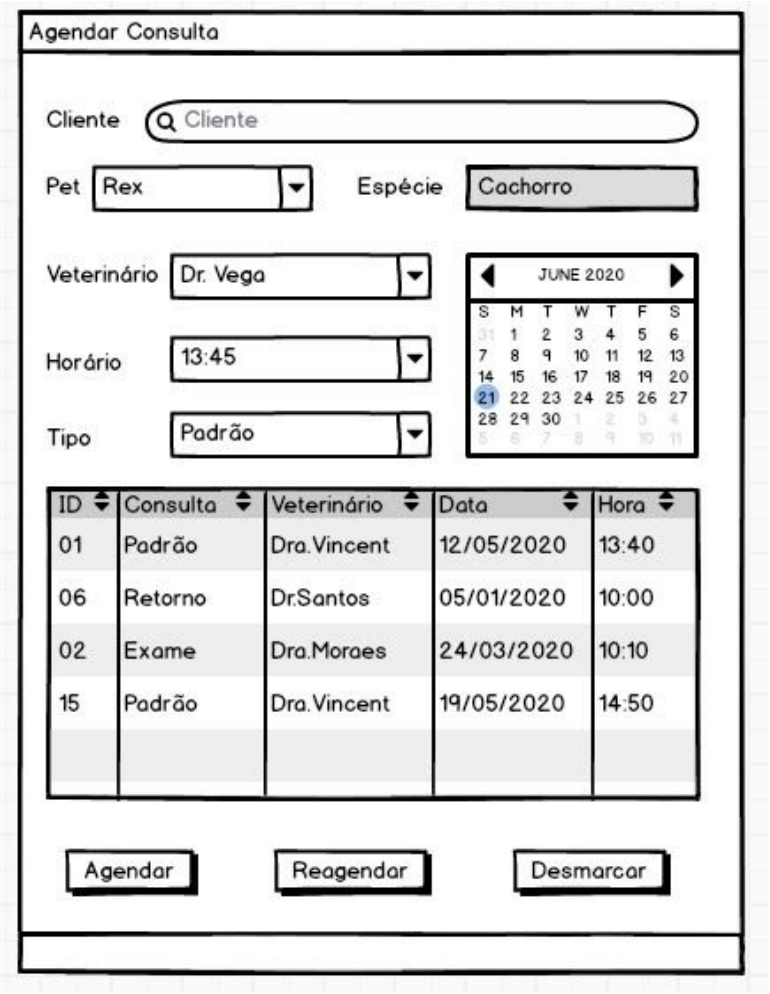
10- Modele uma VCP para o CSU03, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos. Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU03.



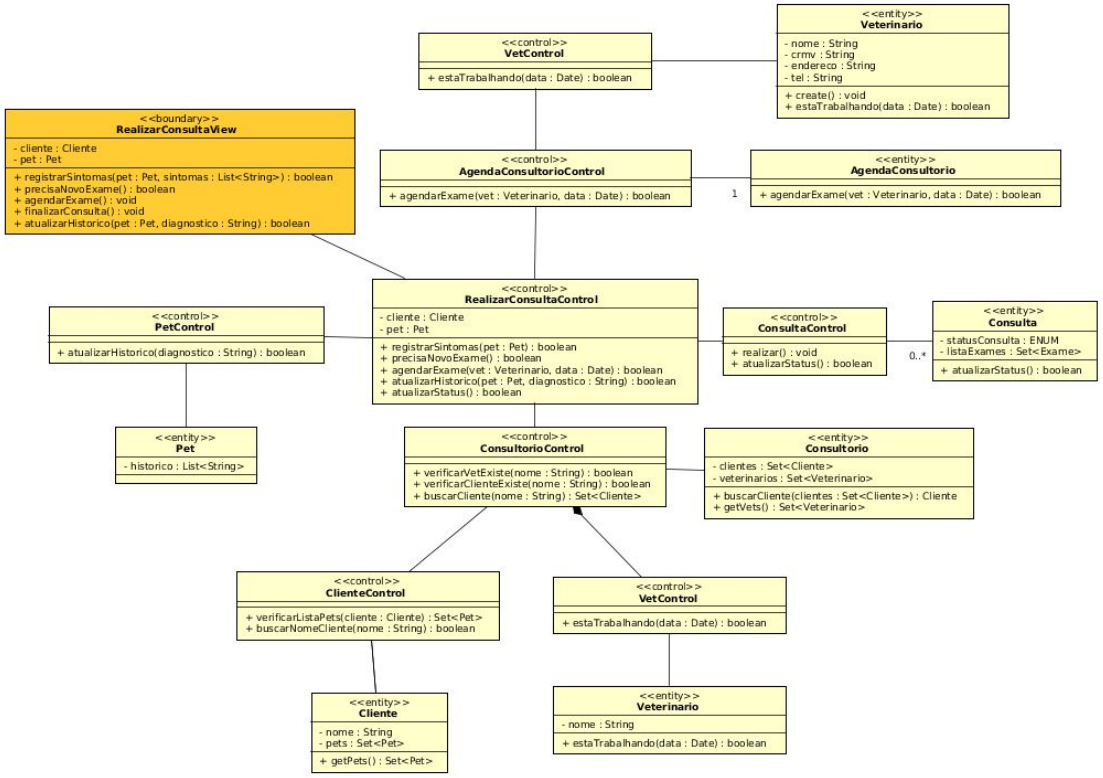


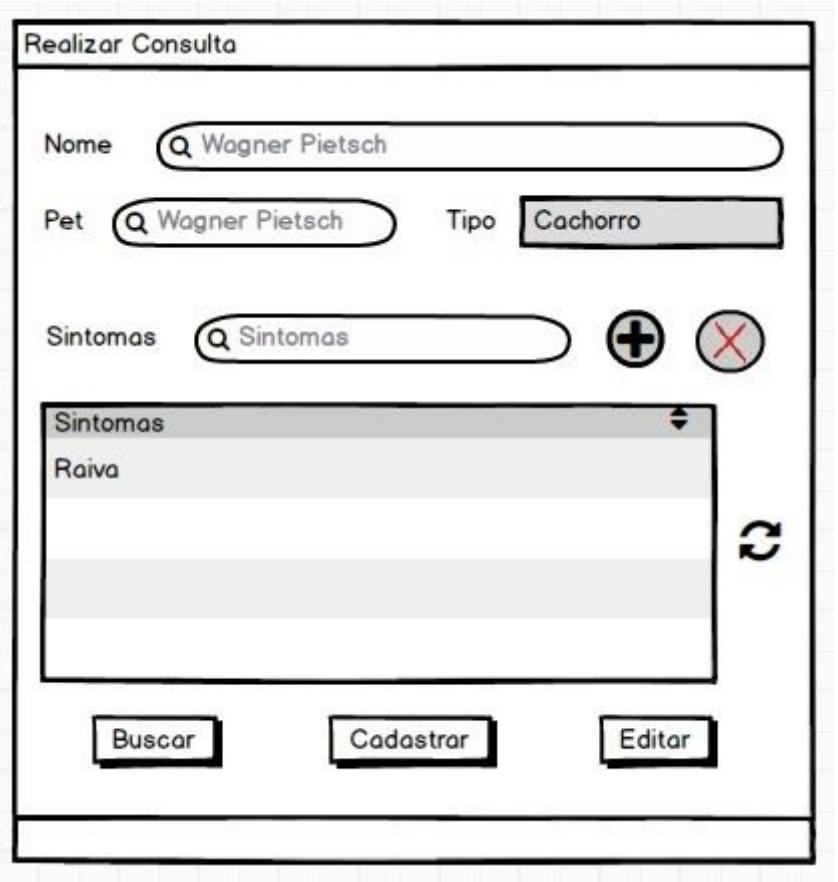
11- Modele uma VCP para o CSU04, utilizando a categorização BCE. Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU04.



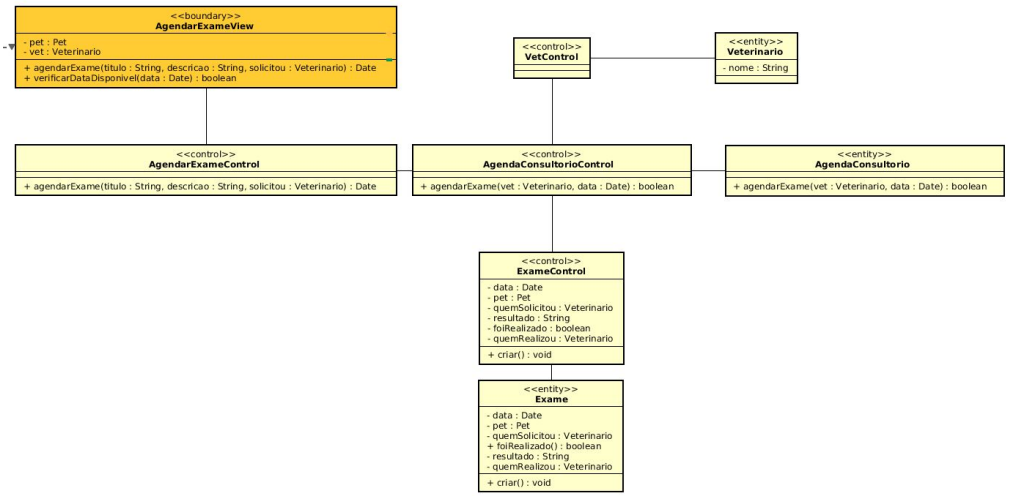


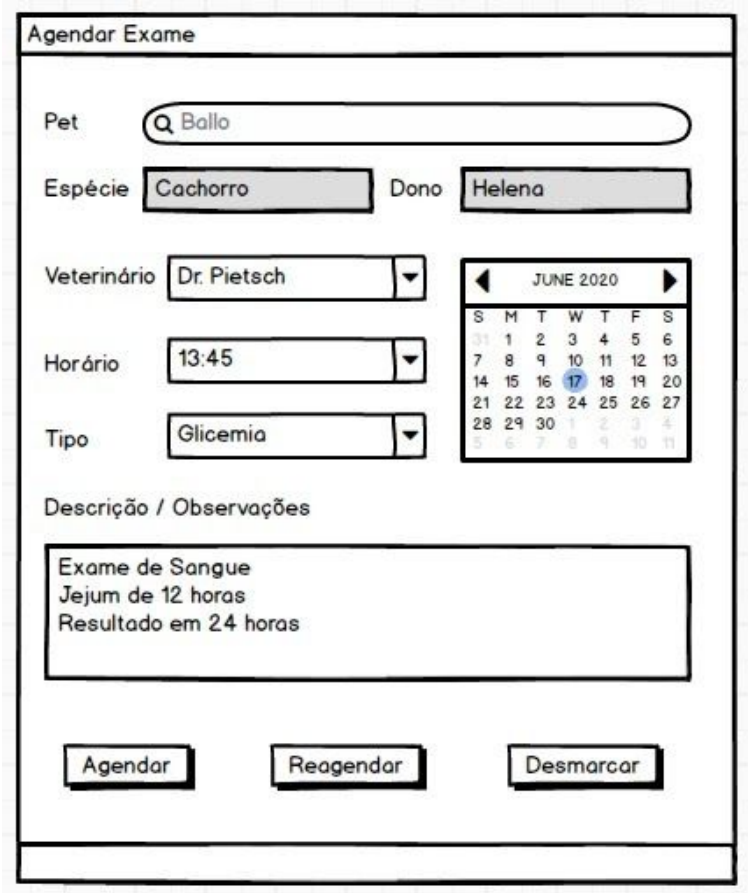
12- Modele uma VCP para o CSU05, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos. Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU05.


<<control>>
VetControl
+ estaTrabalhando(data : Date) : boolean
<<entity>>
Veterinario
- nome : String
- crmv : String
- endereco : String
- tel : String
+ create() : void
+ estaTrabalhando(data : Date) : boolean
<<boundary>>
RealizarConsultaView
- cliente : Cliente
- pet : Pet
+ registrarSintomas(pet : Pet, sintomas : List<String>) : boolean
+ precisaNovoExame() : boolean
+ agendarExame() : void
+ finalizarConsulta() : void
+ atualizarHistorico(pet : Pet, diagnostico : String) : boolean
<<control>>
AgendaConsultorioControl
+ agendarExame(vet : Veterinario, data : Date) : boolean
1
<<entity>>
AgendaConsultorio
+ agendarExame(vet : Veterinario, data : Date) : boolean
<<control>>
PetControl
+ atualizarHistorico(diagnostico : String) : boolean
<<control>>
RealizarConsultaControl
- cliente : Cliente
- pet : Pet
+ registrarSintomas(pet : Pet) : boolean
+ precisaNovoExame() : boolean
+ agendarExame(vet : Veterinario, data : Date) : boolean
+ atualizarHistorico(pet : Pet, diagnostico : String) : boolean
+ atualizarStatus() : boolean
<<control>>
ConsultaControl
+ realizar() : void
+ atualizarStatus() : boolean
0..*
<<entity>>
Consulta
- statusConsulta : ENUM
- listaExames : Set<Exame>
+ atualizarStatus() : boolean
<<entity>>
Pet
- historico : List<String>
<<control>>
ConsultorioControl
+ verificarVetExiste(nome : String) : boolean
+ verificarClienteExiste(nome : String) : boolean
+ buscarCliente(nome : String) : Set<Cliente>
<<entity>>
Consultorio
- clientes : Set<Cliente>
- veterinarios : Set<Veterinario>
+ buscarCliente(clientes : Set<Cliente>) : Cliente
+ getVets() : Set<Veterinario>
<<control>>
ClienteControl
+ verificarListaPets(cliente : Cliente) : Set<Pet>
+ buscarNomeCliente(nome : String) : boolean
<<control>>
VetControl
+ estaTrabalhando(data : Date) : boolean
<<entity>>
Cliente
- nome : String
- pets : Set<Pet>
+ getPets() : Set<Pet>
<<entity>>
Veterinario
- nome : String
+ estaTrabalhando(data : Date) : boolean

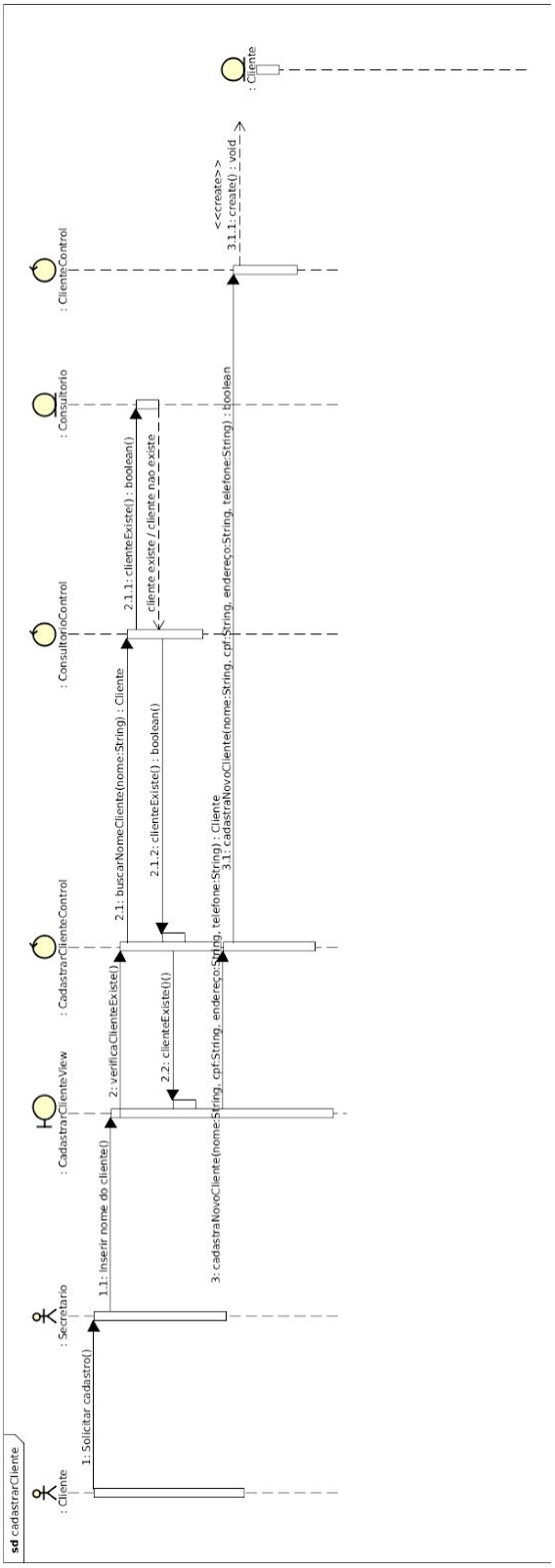

Realizar Consulta
Nome
Wagner Pietsch
Pet
Wagner Pietsch
Tipo
Cachorro
Sintomas
Sintomas
Sintomas
Raiva
Buscar
Cadastrar
Editar

13- Modele uma VCP para o CSU06, utilizando a categorização BCE. A classe de controle deve apresentar dois métodos no mínimo e as classes de entidade devem apresentar no mínimo quatro atributos e dois métodos. Faça também o protótipo de interface de usuário para a classe <<boundary>> do CSU006.

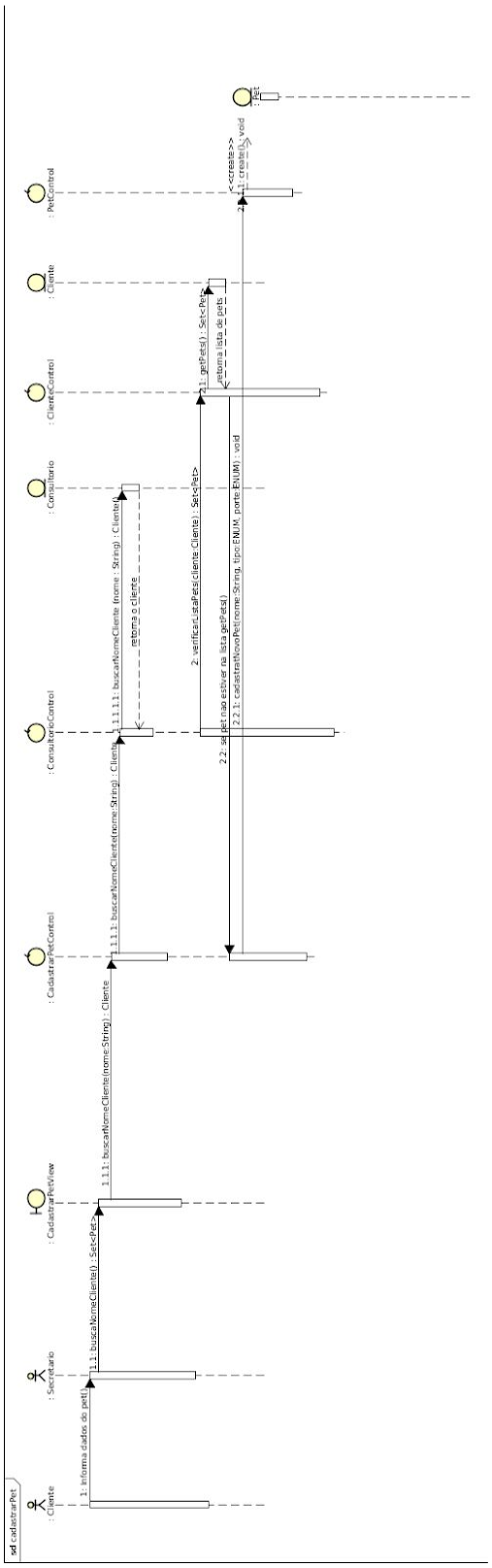




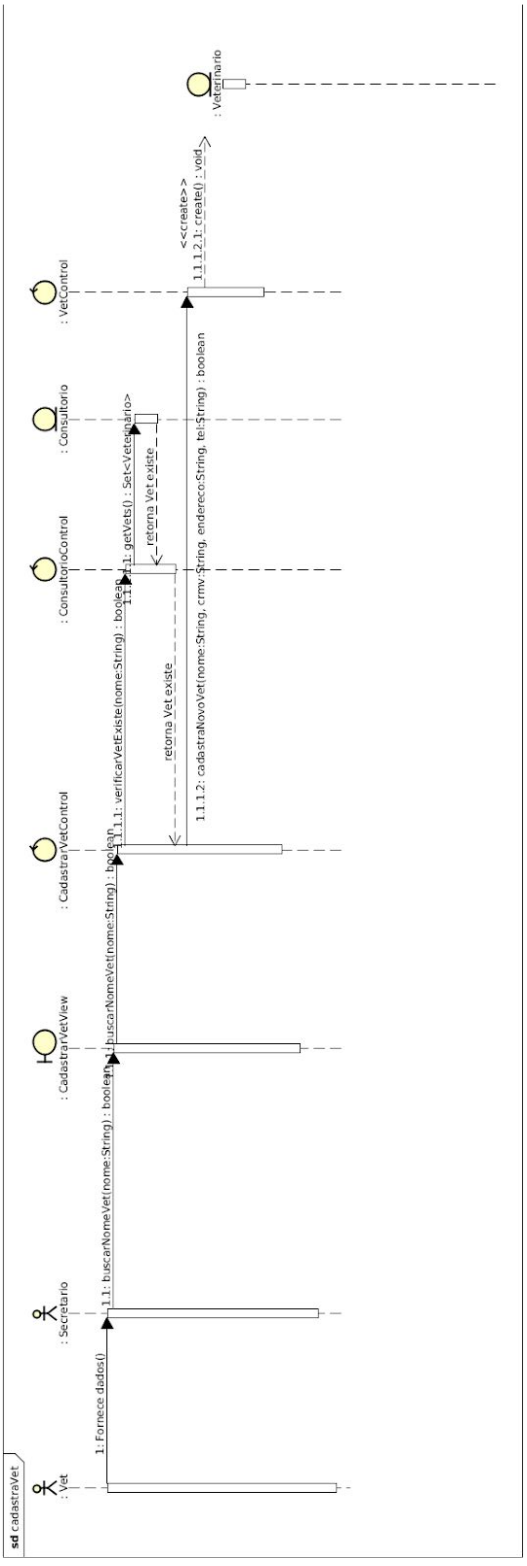
14- Modele um diagrama de sequência para o CSU01. Se a interação for complexa, definam os quadros de interação.



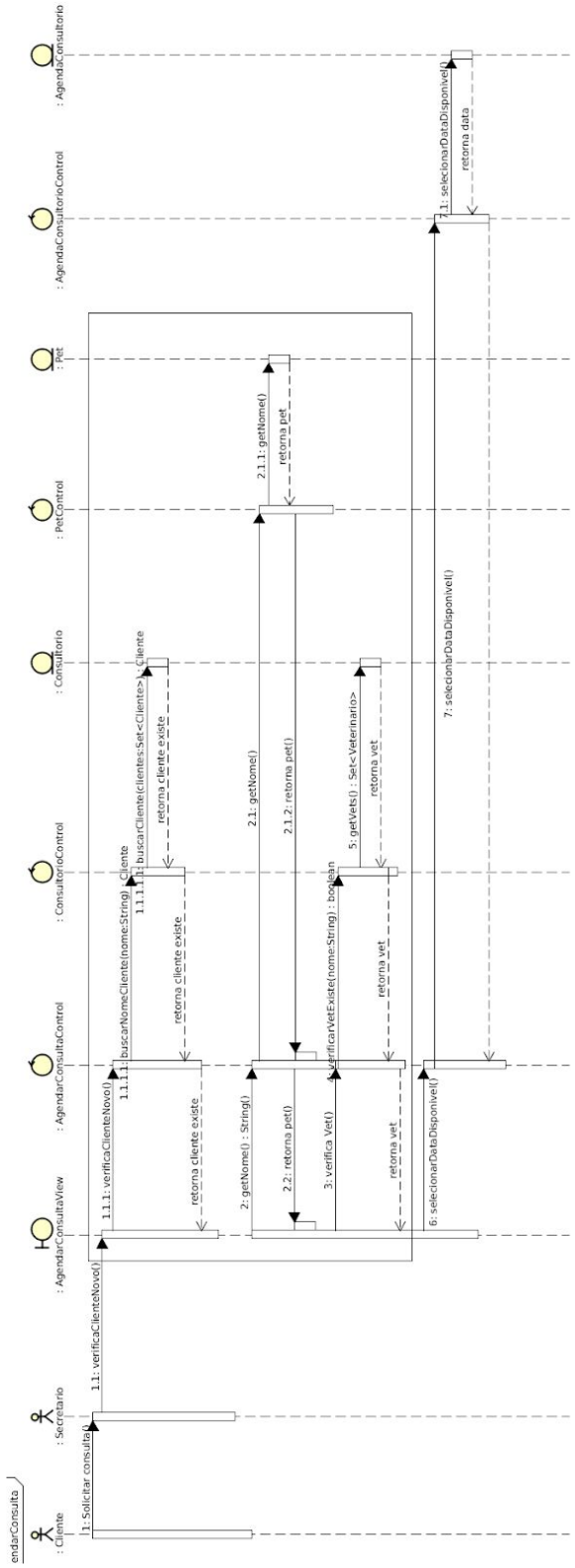
15- Modele um diagrama de sequência para o CSU02. Se a interação for complexa, definam os quadros de interação.



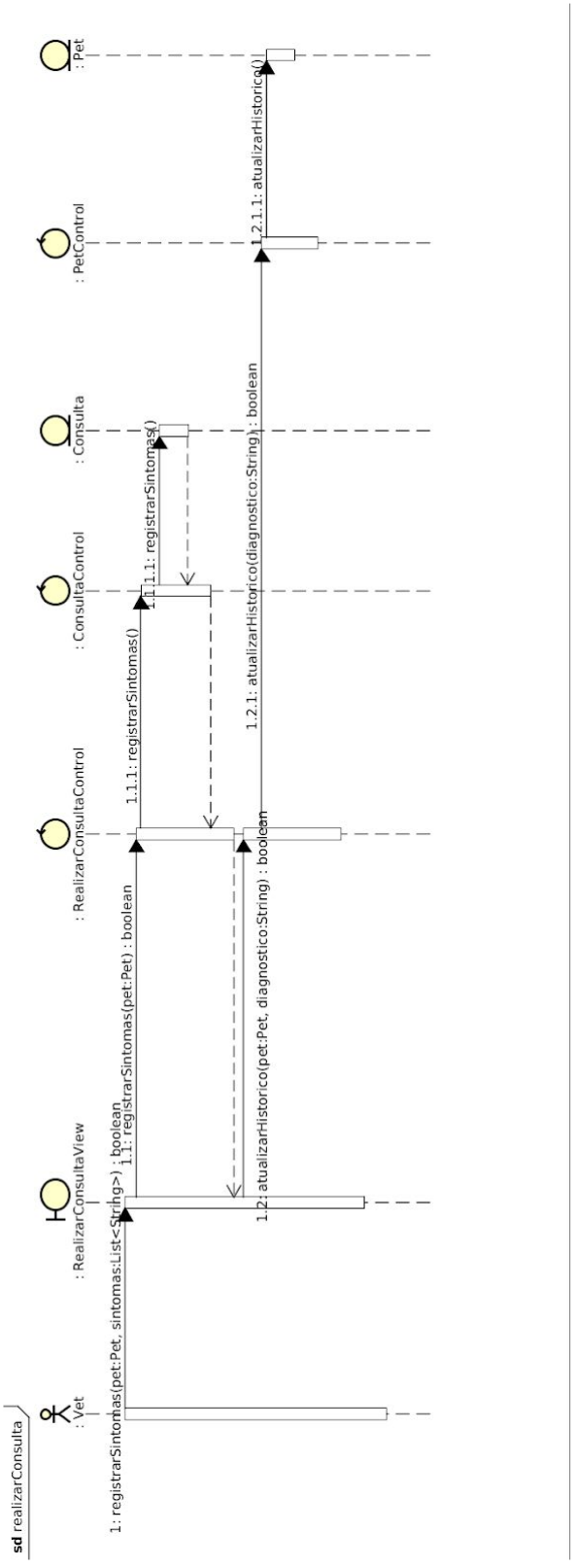
16- Modele um diagrama de sequência para o CSU03. Se a interação for complexa, definam os quadros de interação.



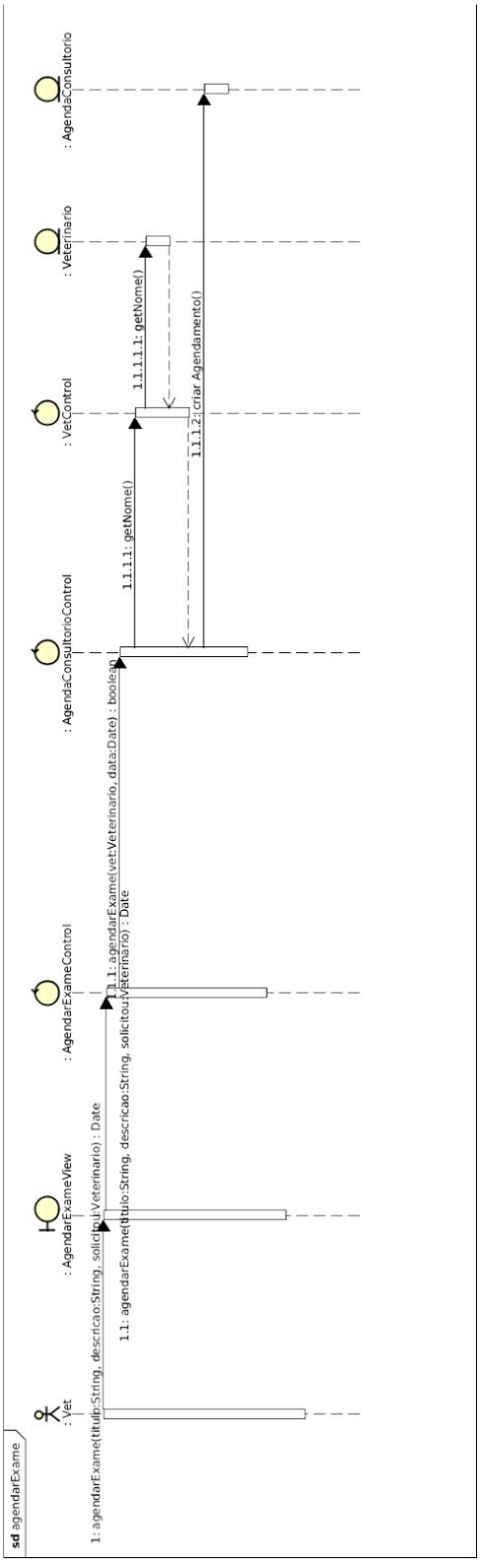
17- Modele um diagrama de sequência para o CSU04. Se a interação for complexa, definam os quadros de interação.



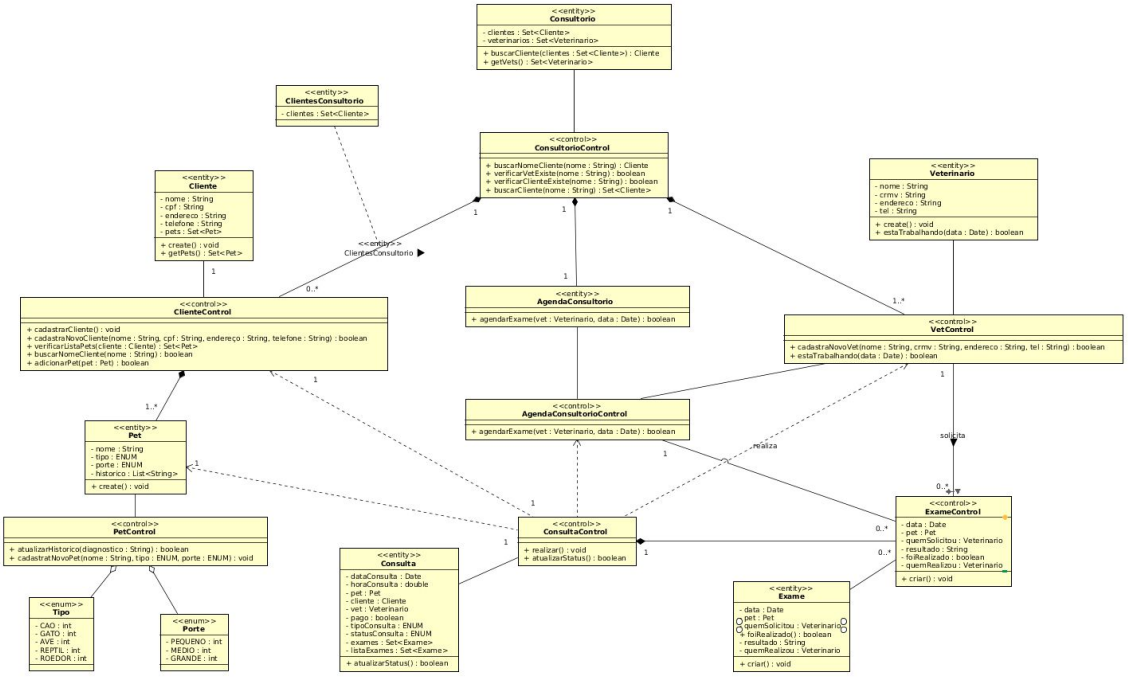
18- Modele um diagrama de sequência para o CSU05. Se a interação for complexa, definam os quadros de interação.



19- Modele um diagrama de sequência para o CSU06. Se a interação for complexa, definam os quadros de interação.



20- Modele um Diagrama de Classes de Projeto a partir das VCPs modeladas e dos diagramas de sequência, e mantenha a utilização da categorização BCE. Os devidos atributos e métodos devem continuar sendo exibidos. As multiplicidades dos relacionamentos devem ser apresentadas.



21- Qual é a classe de entidade mais coesa e a menos coesa do diagrama de classes de projeto? Justifique a tua resposta.

As classes mais coesas são aquelas que possuem pouca ou nenhuma dependência de outras classes. Logo, a mais coesa do nosso diagrama é Veterinario, que não depende de nenhuma informação de outras classes para persistir.

Já a classes menos coesa é Consulta, que depende de Pet, Veterinario, Exame e Cliente.

22- Qual é a classe de entidade mais acoplada e a menos acoplada do diagrama de classes de projeto? Justifique a tua resposta.

A classe mais acoplada é Pet, pois é excluída caso o Cliente também seja excluído, ou seja, é totalmente dependente.

A classe menos acoplada é Veterinário, que não depende de nenhuma outra parte do sistema para ser implementada. É totalmente independente em sua criação.

23- Modele duas relações de gen/espec e ative o princípio de polimorfismo universal de inclusão em cada uma delas. Justifique a razão de existência de cada gen/espec e das operações polimórficas.

Esse modelo, no qual há o princípio de polimorfismo universal, permite que um objeto pertença a várias classes ao mesmo tempo, ou seja, a classe pai aponta para um objeto da classe filha e permite que essa seja instanciada como uma ou como outra.

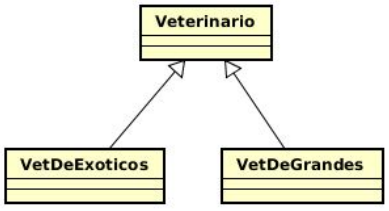
class Consulta{}

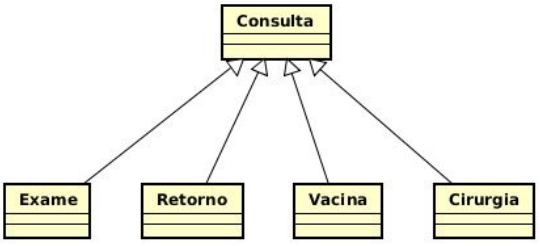
class Exame extends Consulta{}

class Retorno extends Consulta{}

Consulta c1 = new Exame();

Consulta c2 = new Retorno();





24 – Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as relações de gen/espec e as operações polimórficas.

```java
package entities;

import java.util.Date;

public class Exame extends Consulta{

    private Date data;

    private Pet pet;

    private Veterinario quemSolicitou;

    public boolean foiRealizado() {
        return false;
    }

    private String resultado;

    private Veterinario quemRealizou;

    public void criar() {

    }

}
```

```java
Consulta c1 = new Exame();
Consulta c2= new Consulta();
```

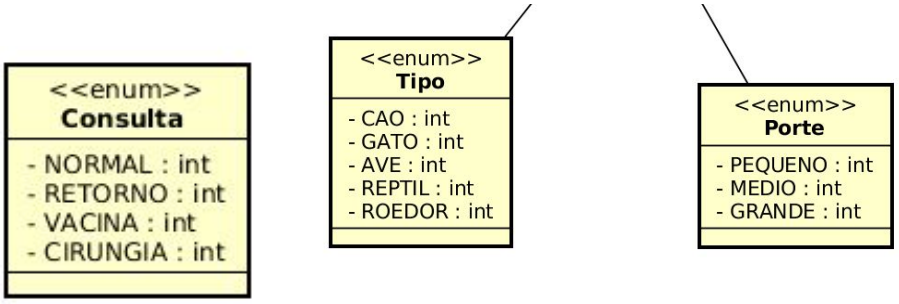
25- As relações de gen/espec modeladas violam o Princípio de Liskov? Justifique a tua resposta.

Não. Todas as classes descritas podem ser substituídas pela sua superclasse.

26- Quais restrições {OCL} sobre gen/espec são aplicáveis nas relações modeladas? Justifique a tua resposta.

A herança quanto ao tipo da Consulta pode ser descrita como incompleta e disjunta, pois dentro do escopo do projeto, temos declarados os tipos de consulta que oferecemos em nossa clínica, mas não abrange todos os tipos de consulta, por exemplo, da medicina humana.

27- Modele três classes enumeradas e utilize as mesmas como tipos de atributos. Justifique a existência de cada uma das classes enumeradas modeladas.



28- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as três classes enumeradas.

```java
package entities.enums;

public enum Porte {

    PEQUENO ('P'),

    MEDIO ('M'),

    GRANDE('G');

    private char numero;

    private Porte(char numero) {
        this.numero = numero;
    }

    public char getPorte() {
        return numero;
    }
}
```

```java
package entities.enums;

public enum Tipo {

    CAO,

    GATO,

    AVE,

    REPTIL,

    ROEDOR;
}
```

```java
public enum Consulta {
    NORMAL('N'),
    RETORNO('R'),
    VACINA('V'),
    CIRURGIA('C');

    private char letra;

    private Consulta(char letra) {
        this.letra = letra;
    }

    public char getConsulta() {
        return letra;
    }
}
```

29- Modele seis membros estáticos, sendo três atributos e três métodos. Justifique a criação de existência de cada um dos membros estáticos modelados.

Os métodos e atributos estáticos são aqueles que não têm seu conteúdo alterado durante a execução do programa. Se ele tem uma função específica, como zerar o valor de uma variável, vai permanecer eternamente com essa atribuição.

Métodos que podem ser estáticos, dentro do escopo do Sistema de Clínica Veterinária:

SistemaDePagamento.emitirBoleto();

Cliente.zerarConsultasAcumuladas();

Cliente.obterNumConsultas();

Atributos que podem ser estáticos, dentro do escopo do Sistema de Clínica Veterinária:

Cliente = static int numConsultas;

Veterinario = static String crmv;

Pet = static int contConsultas;

30- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar os seis membros estáticos.

```java
public class Cliente {

    static int numConsultas;


    public static int obterNumConsultas() {
        return numConsultas;
    }

    public static void zerarConsultasAcumuladas(int numConsultas) {
        numConsultas = numConsultas*0;
    }
}
```

```java
public class Pet {

    static int contConsultas;
}
```

```java
public class SistemaDePagamento {

    public static void emitirBoleto() {
        System.out.println("boleto gerado com sucesso!");
    }
}
```

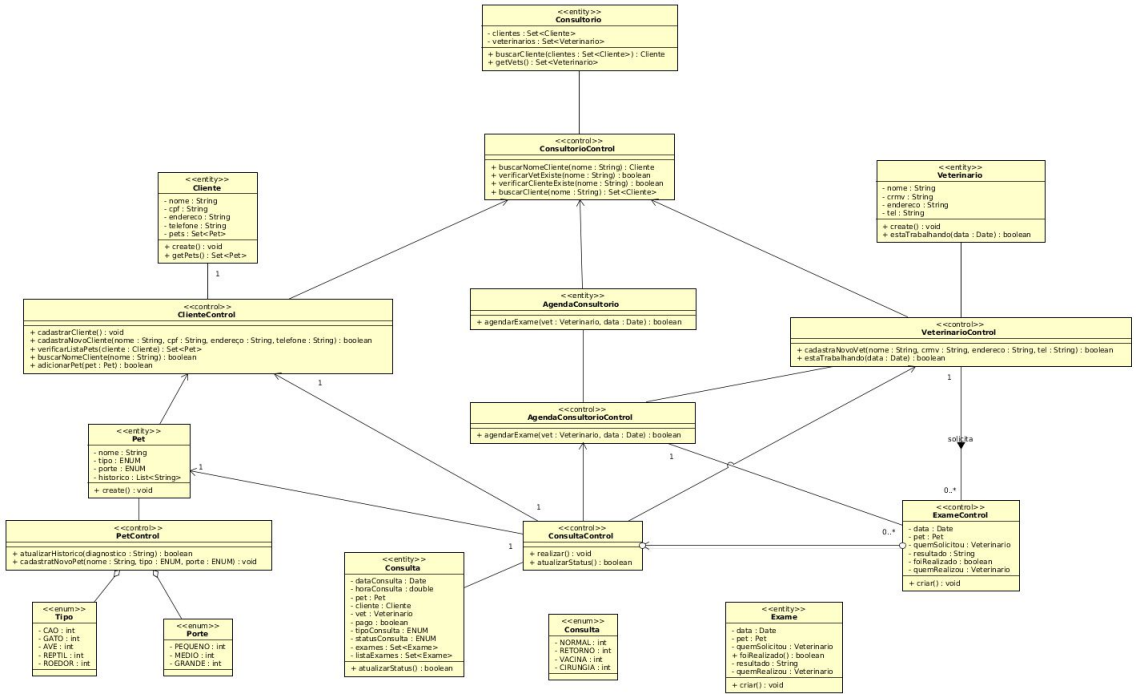
```java
public class Veterinario {

    static String crmv;
}
```

31- Transforme todos os relacionamentos de associação ou agregação entre as classes de entidade e todos os relacionamentos entre as classes de fronteira e controle para dependências estruturais. Explique a vantagem e desvantagem desse tipo de dependência.

A dependência estrutural é um tipo de dependência mais fácil de ser implementado e pensado. Nessa dependência, a classe dependente possui um atribuo que referencia outra classe.

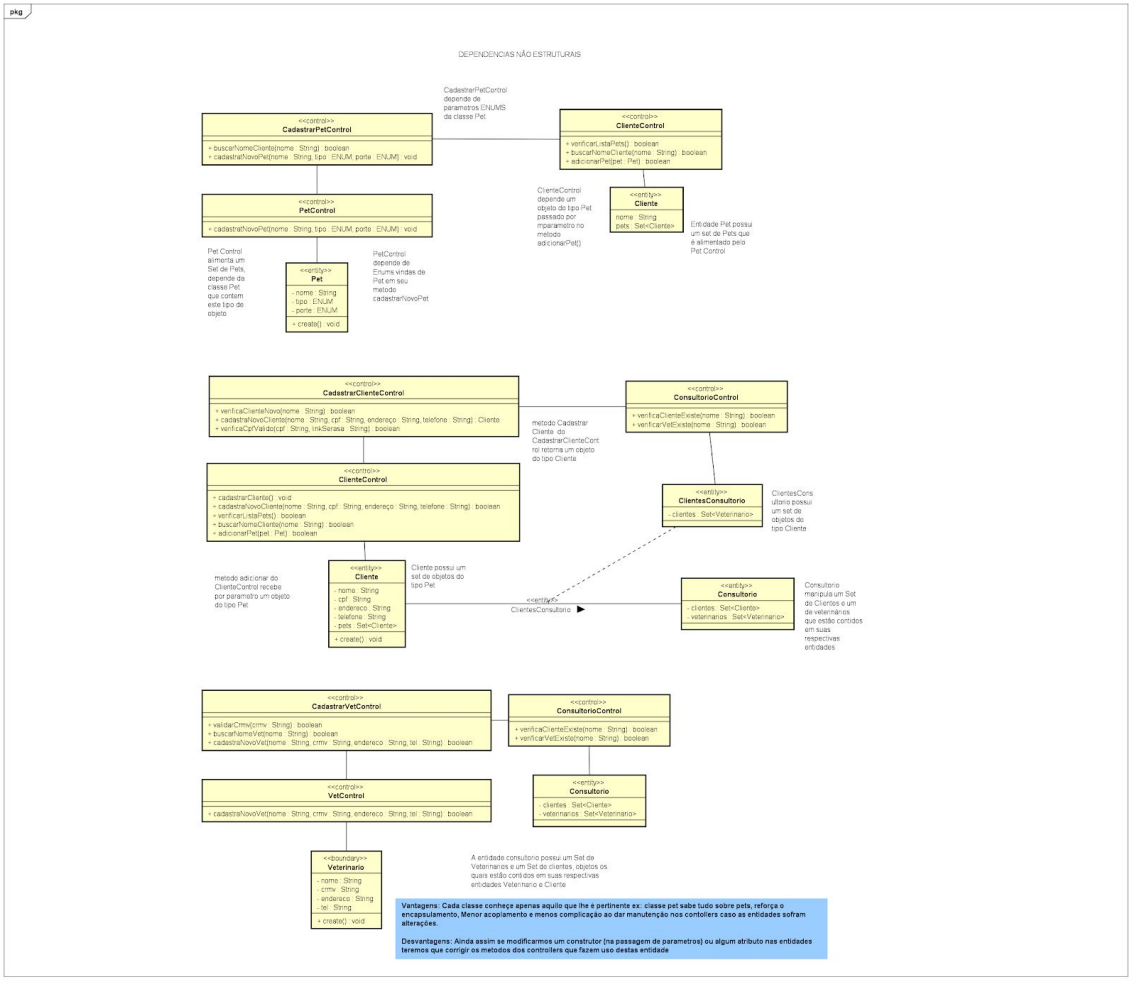
Isso acaba ocasionando um alto acoplamento (baixa coesão). Por isso, precisamos observar se as dependências das classes não podem ser modificadas para dependências não estruturais.



32- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as dependências estruturais.

Classes em java estão anexas também.

33- Modele concomitantemente as dependências não estruturais por parâmetro e por variável local entre as classes de controle e entidade. Explique a vantagem e desvantagem desse tipo de dependência.



34- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as dependências não estruturais por parâmetro e por variável local.

```java
package classeParametrizada;

import java.util.ArrayList;
import java.util.List;

public class ClasseListExames <E> {

	private List<E> listaList;


	public ClasseListExames() {
		listaList = new ArrayList<E>();
	}


	public E add(E e) {
		if (listaList.add(e))
		  return e;
		else
		  return null;
	}

	public int size() {
	      return listaList.size();
	}

	public void clear() {
		listaList.clear();
	}

}
```

```java
package classeParametrizada;

import java.util.HashSet;
import java.util.Set;

public class ClasseSetExame <E> {

	private Set<E> listaSet = new HashSet<>();

	public E add(E e) {
	if (listaSet.add(e))
		return e;
	else
		return null;
	}

	public int size() {
		return listaSet.size();
	}

	public void clear() {
		listaSet.clear();
	}

}
```

```java
package classeParametrizada;

public class Exame {

	private int numero;
	private String descricao;
	private String resultado;

	public Exame (int numero, String descricao, String resultado) {
		this.numero = numero;
		this.descricao = descricao;
		this.resultado = resultado;
	}

}
```

```java
public class Principal <T>{


	public static void main(String[] args) {

		Exame e1 = new Exame(1, "Teste", "teste");
		Exame e2 = new Exame(2, "Teste", "teste");


		System.out.println("-------------------------------------------------");
		System.out.println("Lista LIST");
		System.out.println("-------------------------------------------------");
		ClasseListExames<Exame> exameList = new ClasseListExames<Exame>();

		//Adicionando itens
		exameList.add(e1);
		exameList.add(e2);

		System.out.println(exameList.size());

		//Esvaziar lista
		exameList.clear();
		System.out.println(exameList.size());
		//-----------------------------------------------
		System.out.println("-------------------------------------------------");
		System.out.println("Lista SET");
		System.out.println("-------------------------------------------------");

		ClasseSetExame<Exame> exameSet = new ClasseSetExame<Exame>();

		//Adicionando itens
		exameSet.add(e1);
		exameSet.add(e2);

		System.out.println(exameSet.size());

		//Esvaziar lista
		exameSet.clear();

		System.out.println(exameSet.size());
	}

}
```

35- Modele concomitantemente as classes parametrizadas com a estrutura <List> e <Set> para resolver o lado muitos dos relacionamentos. Para cada classe parametrizada modelada, justifique o motivo de ter escolhido o tipo de estrutura de dados.

36- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar as classes parametrizadas com a estrutura <List> e <Set>.

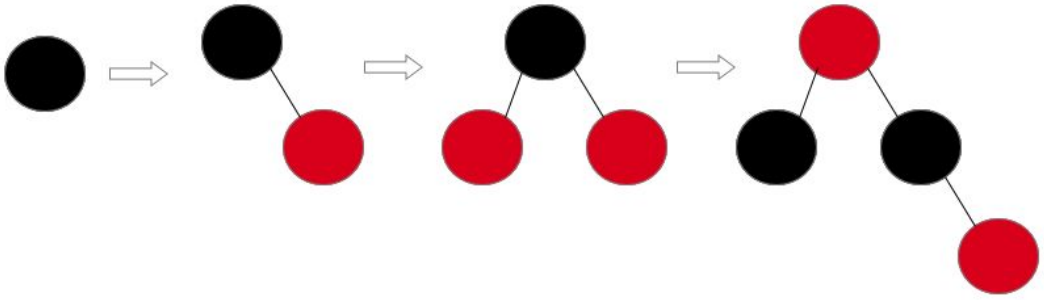
37- Modele a estrutura <TreeSet>. Justifique a razão dessa estrutura no seu diagrama. Represente graficamente como essa árvore trabalharia em tempo de execução.

Estruturas de dados TreeSet utilizam arvores binarias balanceadas Rubro Negras as quais utilizam coloração nos nós para manter o balanceamento. (Árvore binária simétrica)

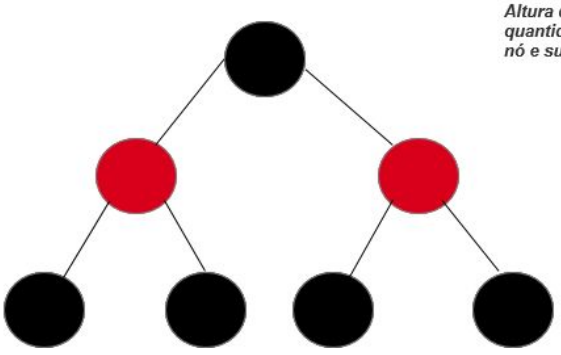
Toda raiz é preta.
Se um nó é vermelho, obrigatoriamente seus filhos são pretos.
Faz balanceamento local, apenas os locais onde houve remoção ao adição são balanceados atraves do ajuste de cores, estas operações corrigem as propriedados da arvore.

Inserção:

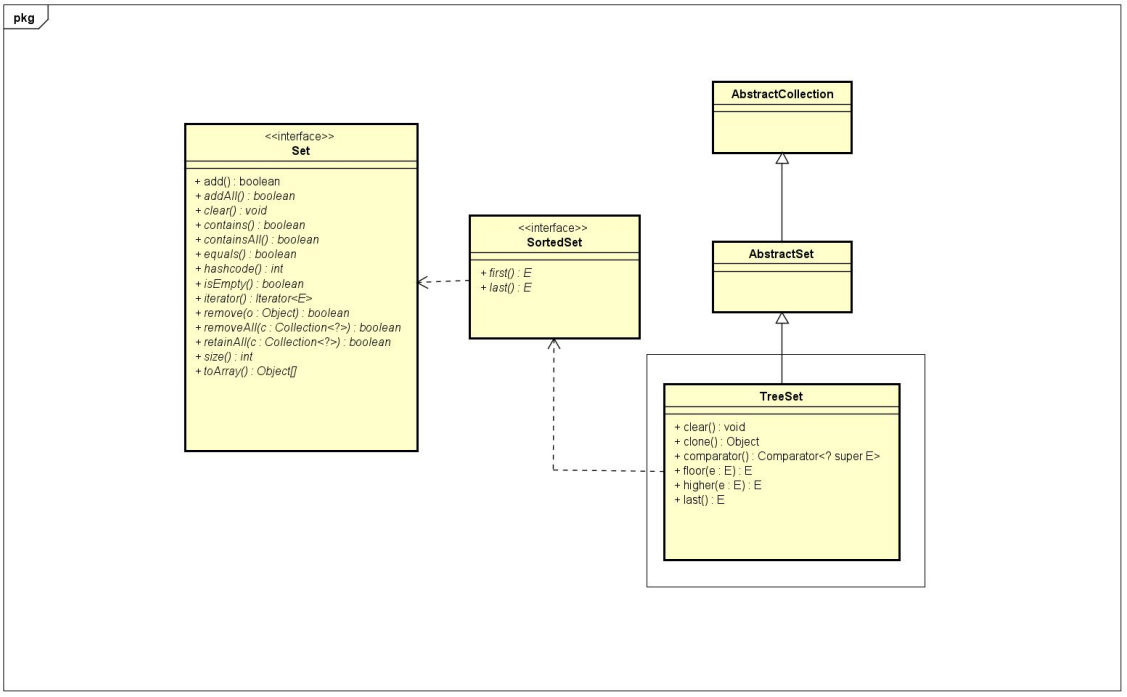
Nós inseridos sempre na cor vermelha



A estrutura da arvore deve serguir sempre este modelo:



Operações:
Inserção
remocao
Altura de um nó (Calculada atravpés da quantidade de nós pretos no caminho entre o nó e sua folha descendente possui).



38- Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar a estrutura de dados <TreeSet>.

```java
package treeSet;

import java.util.Iterator;

public class Principal {

    public static void main(String[] args) {

        //Criando TreeSet
        TreeSet<String> ts = new TreeSet<String>();

        //Adicionando itens
        ts.add("Fernanda Reis");
        ts.add("Bruno Harnik");
        ts.add("Raquel Nascimento");
        ts.add("Luiz Fernando");

        //Exibindo elementos
        Iterator<String> iter = ts.iterator();
        while(iter.hasNext()) {
            System.out.println(iter.next());
        }

        //Exibindo em ordem reversa
        Iterator<String> rev = ts.descendingIterator();
        while(rev.hasNext()) {
            System.out.println(rev.next());
        }
    }
}
```

39- A partir do caça objetos abaixo, encontre no mínimo cinco objetos e modele uma relação todo-parte, abstraindo o nome de uma ave na classe todo e os cinco objetos encontrados como partes deste todo. Explique a solução apresentada.

A classe Ave possui 5 atributos essenciais para ser construída. Foi instanciada como uma galinha.

Class Ave {

String bico;

String asa;

String pena;

```
        String ovo;
        Boolean voa;

        public Ave (String bico, String asa, String pena, String ovo, boolean voa){
        this.bico = bico;
        this.asa = asa;
        this.pena = pena;
        this.ovo = ovo;
        this.voa = voa;
}
Class Natureza {

        public static void main (String [] args) {
                Ave galinha = new Ave("curto", "pequenas", "finas", "pequeno", false);
        }
```

40- Analise o diagrama de classes abaixo. Descreva a coesão e acoplamento desse diagrama. Modele um diagrama de classes que seja melhor em termos de coesão e acoplamento e explique o porquê do novo diagrama de classes ser mais vantajoso do que o abaixo.

Todas as classes desse diagrama apresentam alto acoplamento e baixa coesão. Isso significa dizer que as classes possuem múltiplas dependências umas das outras, então se uma delas for modificada, todo o ciclo vai ser quebrado, pois uma classe sofreu alteração. Para um projeto profissional, de negócios, é uma situação inapropriada. Imagine que está trabalhando em um projeto com mais de 100 classes interdependentes (alto acoplamento) e precisa executar alteração em uma delas. Todo o projeto necessitará de refatoração e isso, numa empresa, envolve tempo e desperdício de tempo é desperdício de

dinheiro. Devemos, então, aumentar a coesão e diminuir o acoplamento para que as classes possam ser modificadas sem maiores influências em um grande número de outras classes.

Para remodelar o diagrama acima, precisaríamos conhecer minimamente o domínio, pois no diagrama não está nem indicado qual é a relação de atributos entre uma classe e outra. A única coisa que se pode concluir é que precisaríamos inserir moderadores de acesso (public, private, protected e default) e, ao menos, uma ou mais classes abstratas ou interfaces.

41 - Identifique cada um dos relacionamentos utilizados no diagrama de classes abaixo, inclusive se são simétricos ou assimétricos. Quais atributos a instância de cada classe pode visualizar? Justifique a tua resposta.

Relacionamentos:

Dependência: Uma classe depende da outra para que toda a sua funcionalidade seja implementada corretamente.

E - G → dependência. E necessita de G.

Composição: Quando o container é destruído, o conteúdo também é.

Então, no caso, as relações de composição presentes entre as classes:

A - F → A possui F. Se A é destruída, F também é

E - B → E possui B. Se B é destruída, E também é.

Generalização/Especialização: representa a herança entre classes.

B - A → Herança - B herda de A.

F - C → Herança - C herda de F.

E - C → Herança - E herda de C.

E - D → Herança. E herda de D.

A classe E possui herança múltipla.

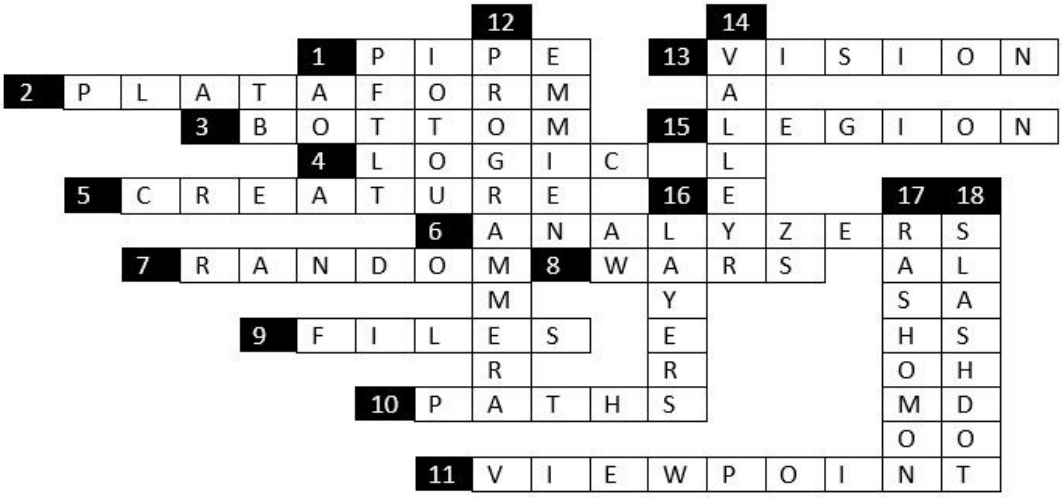
Visibilidade de atributos:

B - A → como B herda de A, B tem permissão para visualizar os atributos protegidos de A.

E - D → como E herda de D, E tem permissão para visualizar os atributos protegidos de D.

Todos os outros atributos de outras classes são visíveis apenas para a própria classe, visto que são atributos privados.

42- Elabore um jogo de palavras cruzadas contemplando 18 termos identificados no artigo.



43- Descreva as características de cada um dos programadores cegos.

Programador 1 – descreve o animal por cima, toca o animal e explica aos outros. Analisa profundamente, coloca o ouvido perto do bicho, análise mais material das coisas.

Programador 2 – descreve o animal mais minuciosamente, observa com mais atenção os detalhes sonoros.

Programador 3 – metódico, calculista, presta muita atenção nos detalhes, nas texturas, abre caixas

Programador 4 – sente cheiros e percebe que tem uma base embaixo do bicho, todas bagunçadas, percebe q a criatura anda em bando

Programador 5 – fina arte, historiadora, de vez em quando artista e psicóloga. Carrega uma escada. Olhar as coisas de cima. O conjunto da obra. Uma mulher de visão. Vê luz e escuridão e formas, de modo meio abstrato.

44- Qual é a relação da estória apresentada com o projeto de desenvolvimento de software?

A conexão diz que é importante ter diferentes pessoas trabalhando em um mesmo projeto, pois cada um tem um ponto de vista e pode identificar melhorias que o outro pode não ter visto.

45- Produza uma charge, a mão, abordando a estória do elefante e dos cinco programadores. Explique o significado da sua charge.

Os participantes da charge, ou do projeto de construção do elefante, não conversaram muito antes de decidir pela melhor opção de como seria o animal e, por isso, criaram algo fantasioso e que não atende tão bem às funcionalidades que deveria.