

Estudo de caso:

# Sistema de Pet Shop

Engenharia de Software e Design Orientado a Objeto

Marcelo Eduardo Alves Paixão Resende

UNIVERSIDADE NORTE DO PARANÁ – UNOPAR  
Uberlândia  
2016

# Objetivo geral

Produzir documentos que auxiliem no desenvolvimento de Softwares de Pet Shop

# Objetivos específicos

- Análise de requisitos
- Diagramas UML: Caso de uso e Classes
- Modelagem do Banco de Dados
- Exemplificação da lógica de programação

Sem um software...  
o que pode dar errado?

Integridade de dados comprometida!

Regras de negócios gerenciada por humanos – falhas!

Menor agilidade na entrega de serviços!

Menor participação do cliente nos processos!

Com um software...  
como pode dar certo?

Integridade de dados e regras de negócios  
reforçadas por código

Organização dos dados é formal

Software agiliza tarefas repetitivas

Cliente PODE participar dos processos

Engenheiros de Software  
**falam na mesma língua**  
*(ou tentam...)*



# Padrões da indústria

## UML

um padrão sólido e reconhecido  
por desenvolvedores

## DER (ERD)

Diagrama de entidade  
-relacionamento, reconhecido  
por desenvolvedores, dbas...

## Hardware/Rede

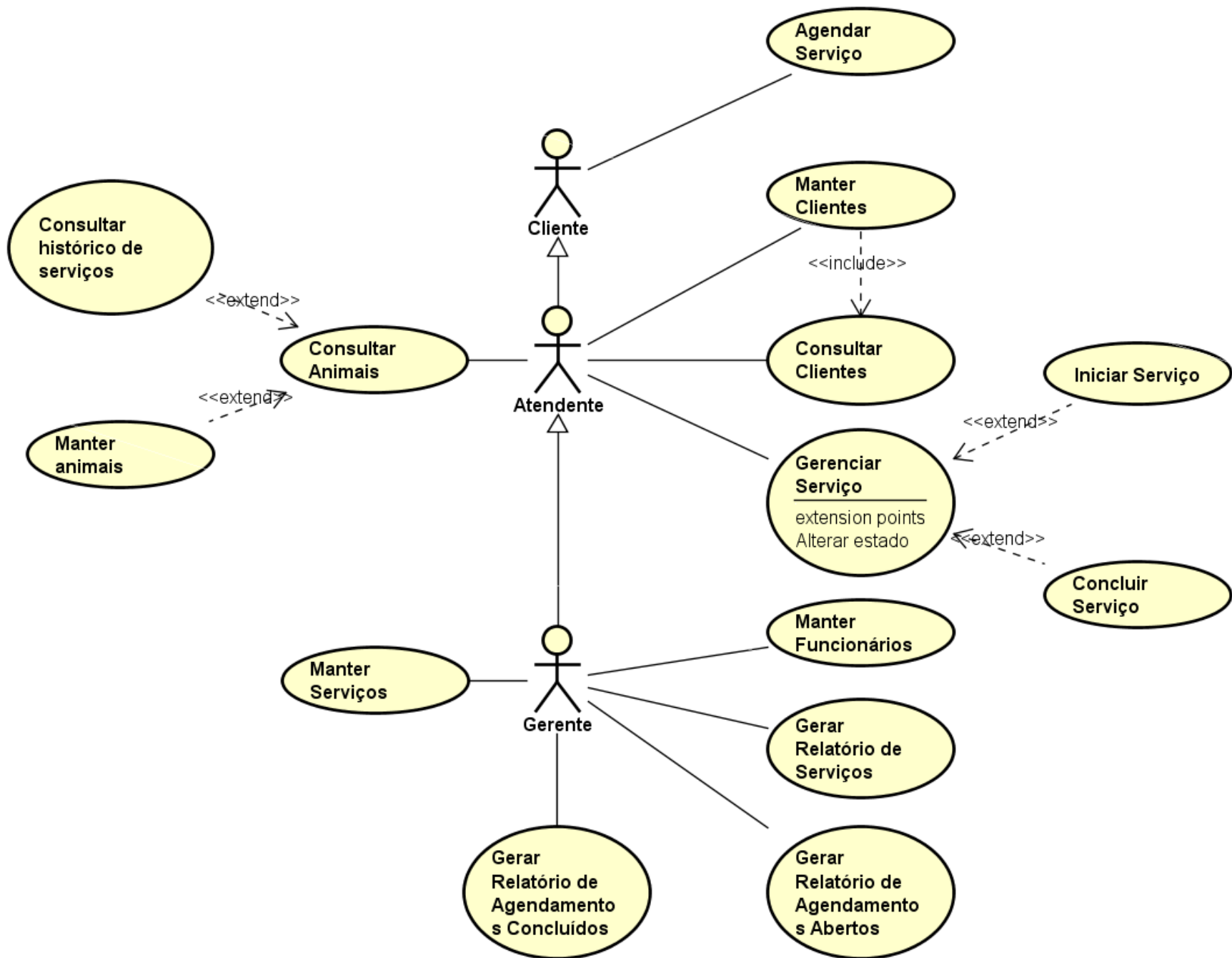
Cisco Packet Tracer permite  
uma representação  
gráfica/simulação

## Design Patterns

Padrões de software orientação  
a objetos

# Diagrama de Casos de uso

- **Devem** ser simples
- O cliente entende diagramas de caso de uso simples
- Guiam a entrega de valor do software
- Guiam metodologias de desenvolvimento (Processo Unificado, FDD)



# Primeiras impressões

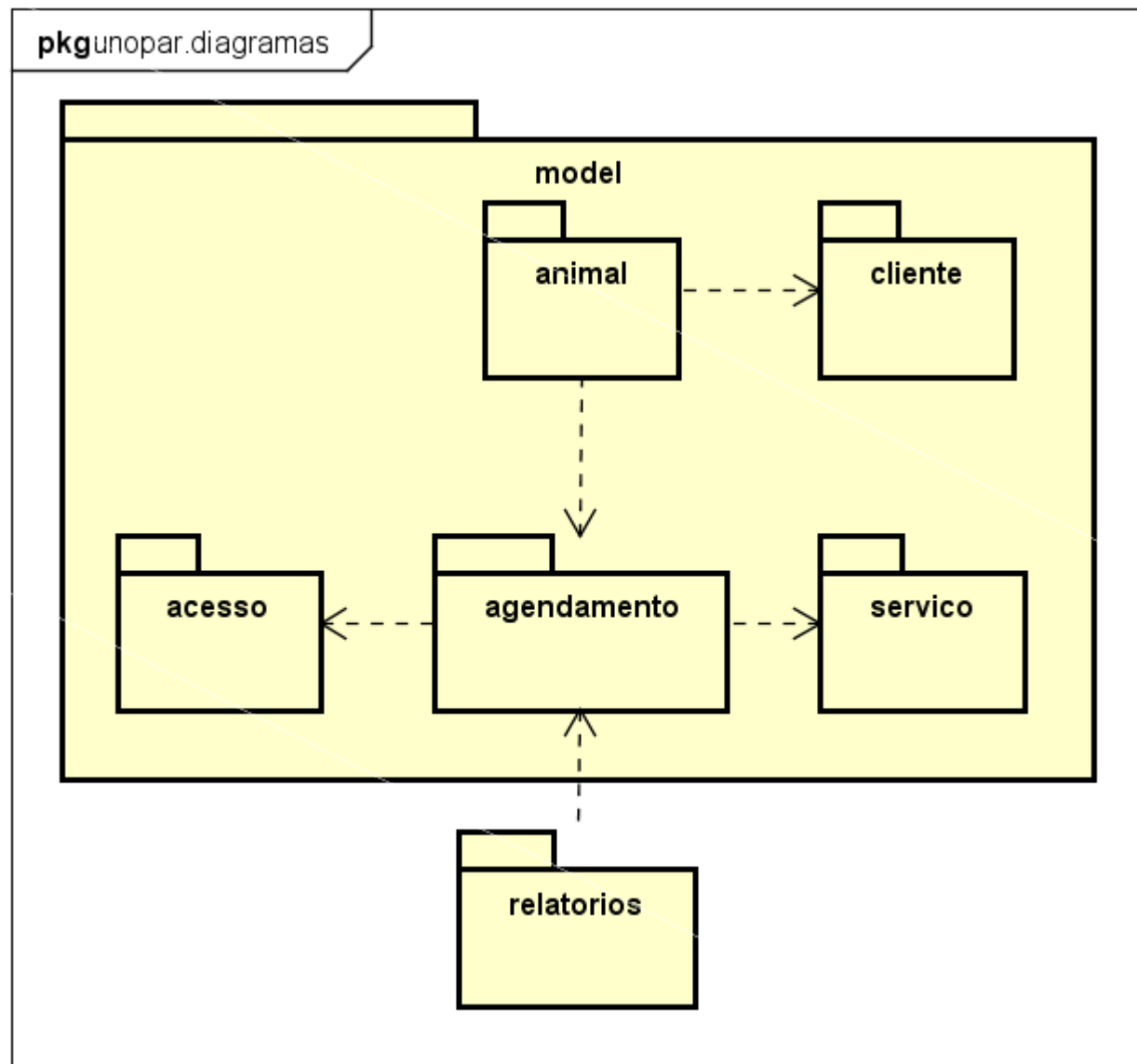
- **Permissões de uso e acesso:** Gerentes, Atendentes, Clientes
- **GUI e UX:** Quais telas são chamadas por quais outras?
- **Tarefas:** Prioridades podem ser atribuídas para guiar o desenvolvimento e iterações

# UML: Diagrama de classes

- Pacotes
- Entidades
- Lógica

# Pacotes

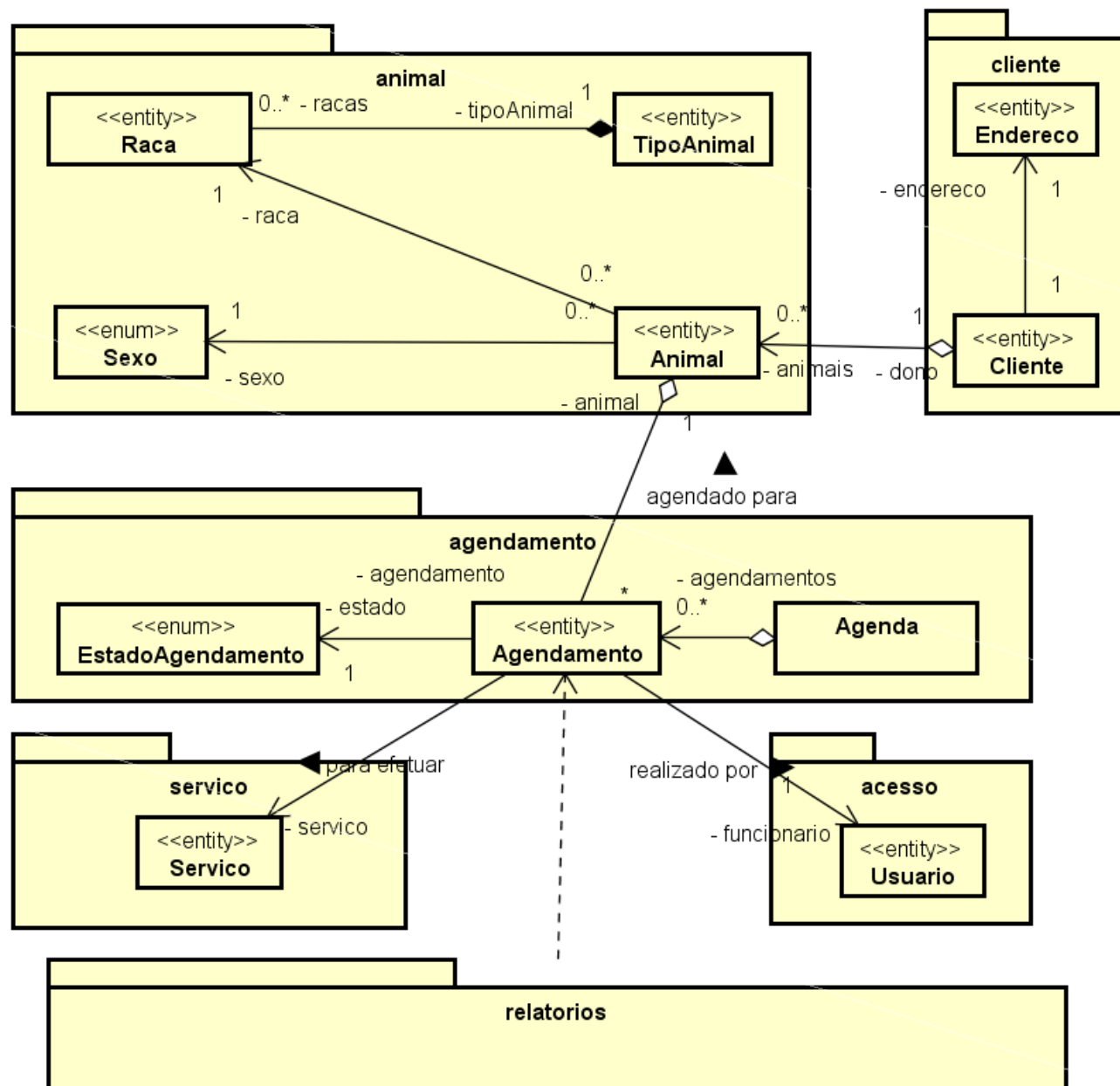
- Encapsulamento
- Organização
- Evita conflitos de nomes
- Agrupas dependências e dependentes

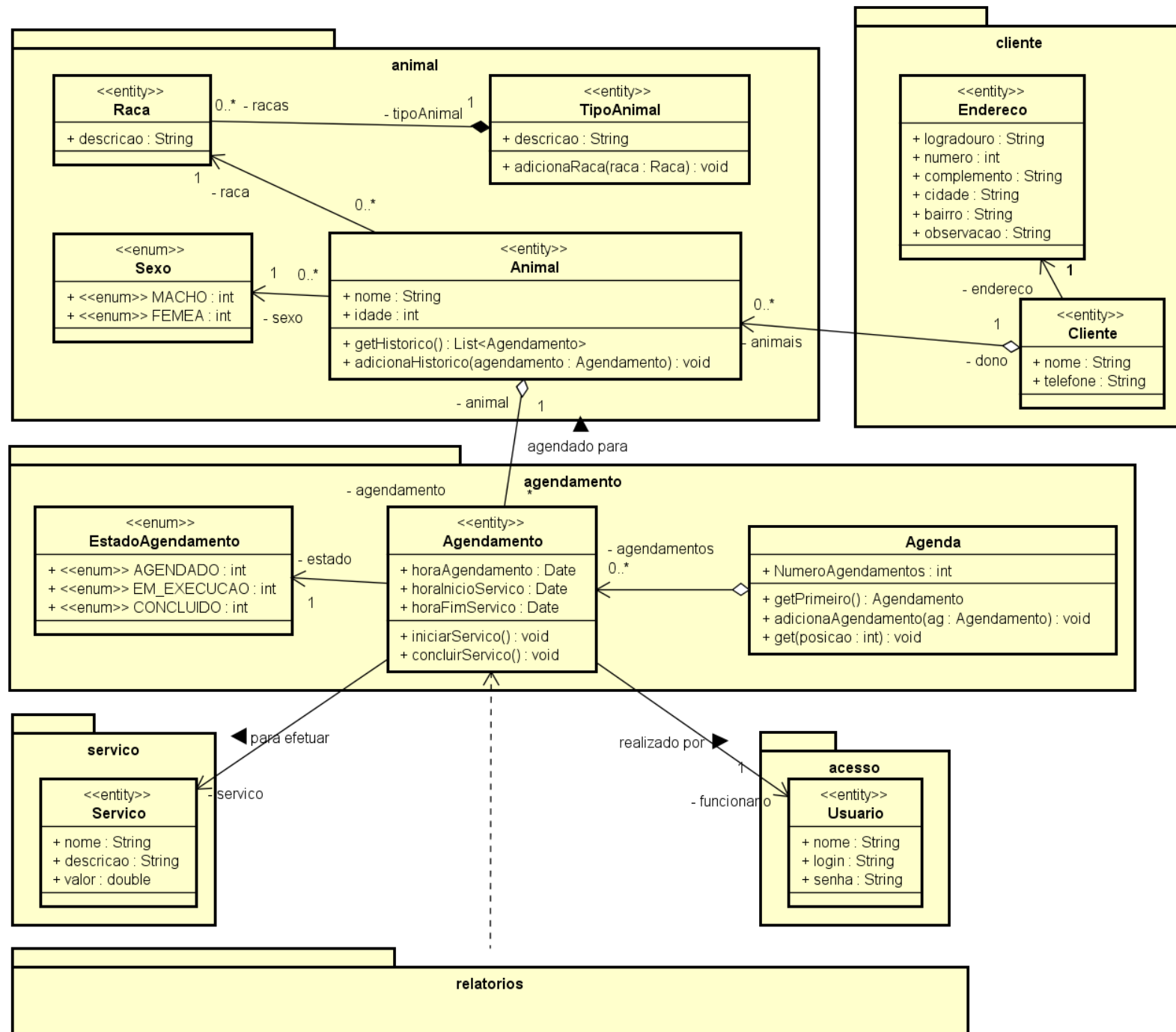


# Diagrama de classes

- Identifica entidades e operações que realizam
- Visão geral da API
- Padrões de projeto

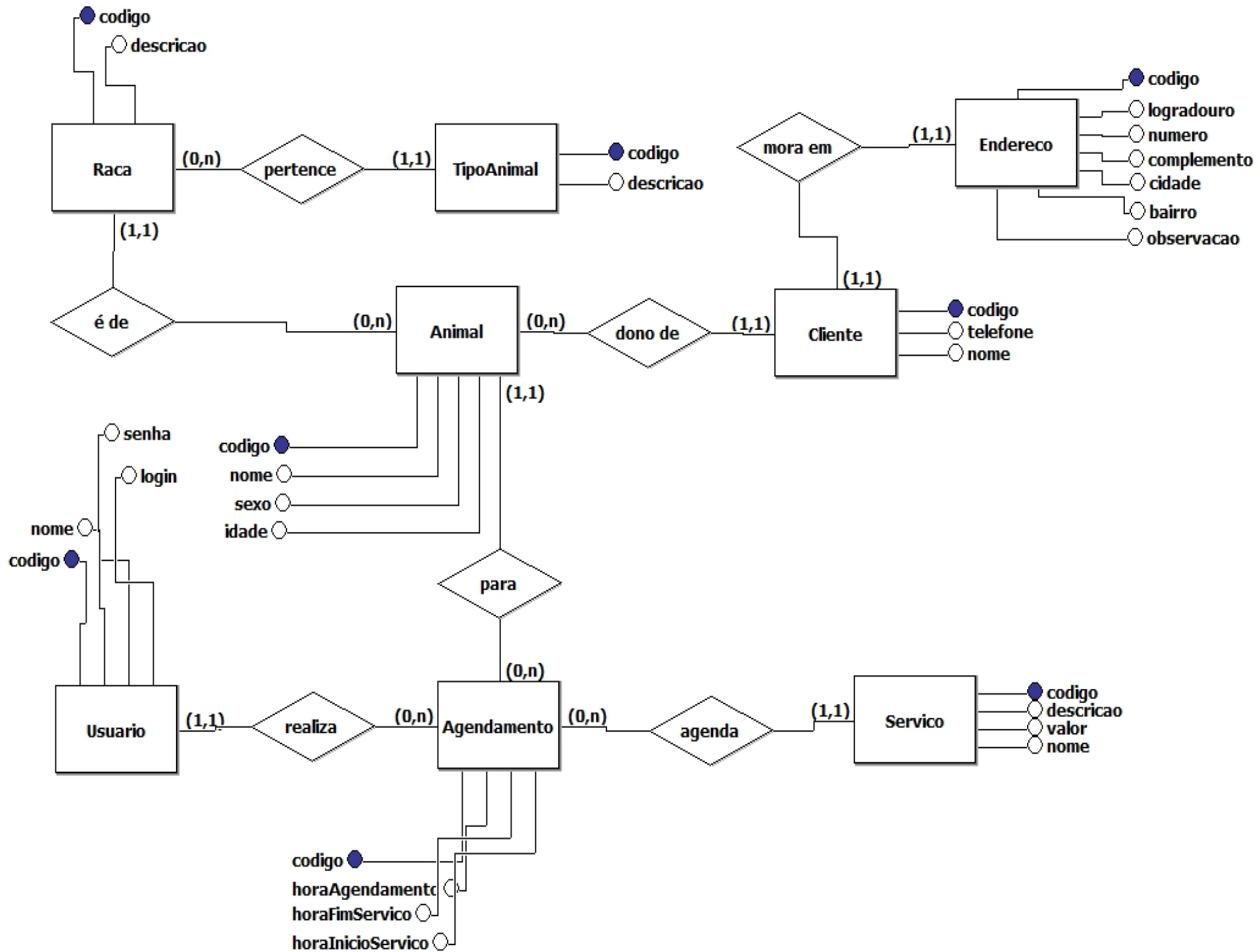


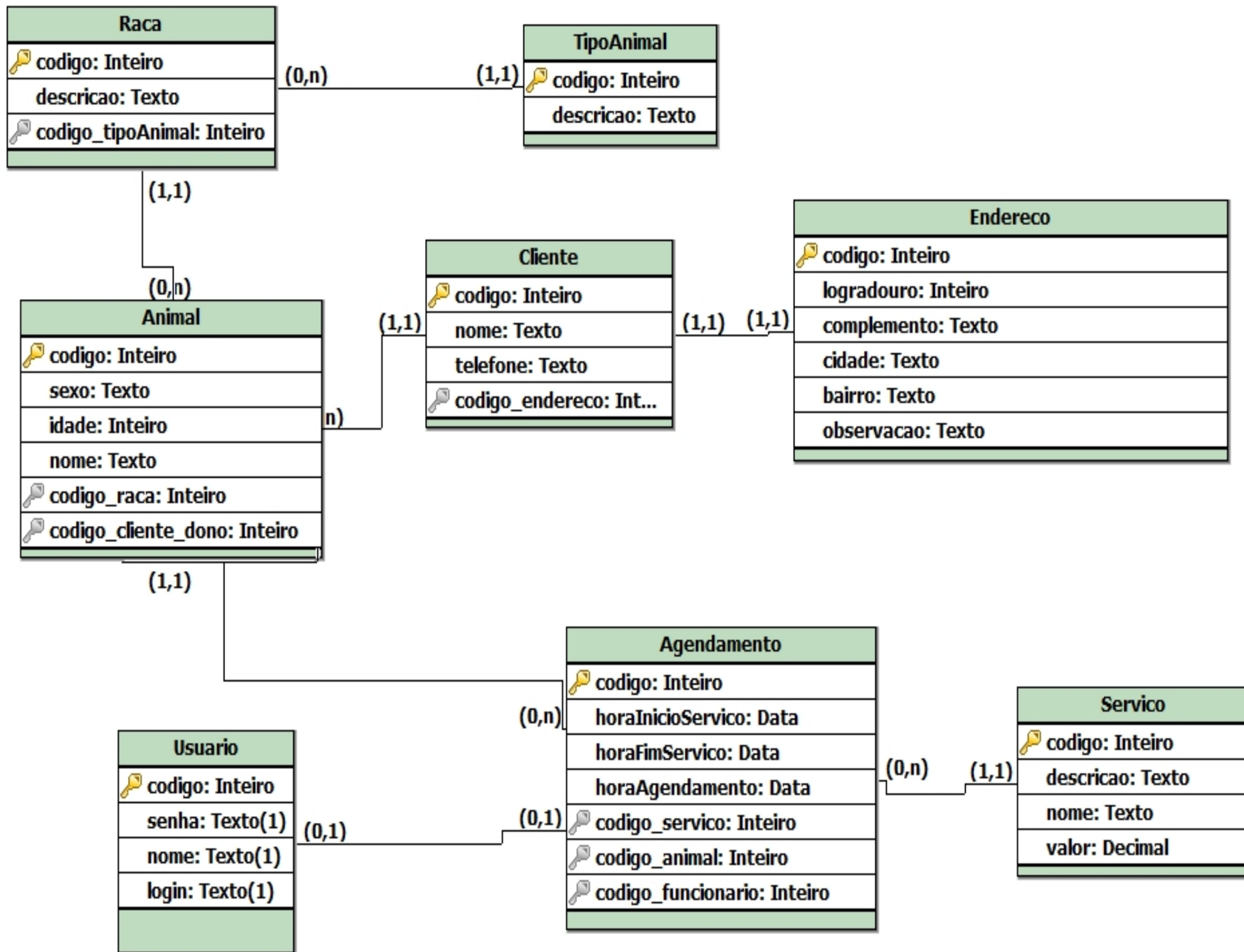




# Modelagem do Banco de Dados

- Visão geral das entidades e dos dados armazenados
- Visão geral da integridade relacional
- Modelo lógico e conceitual independentes da tecnologia final





<Programação />

# Requisitos de uma Agenda

- Ordenada por horário
- Nunca terá agendamentos em horários iguais

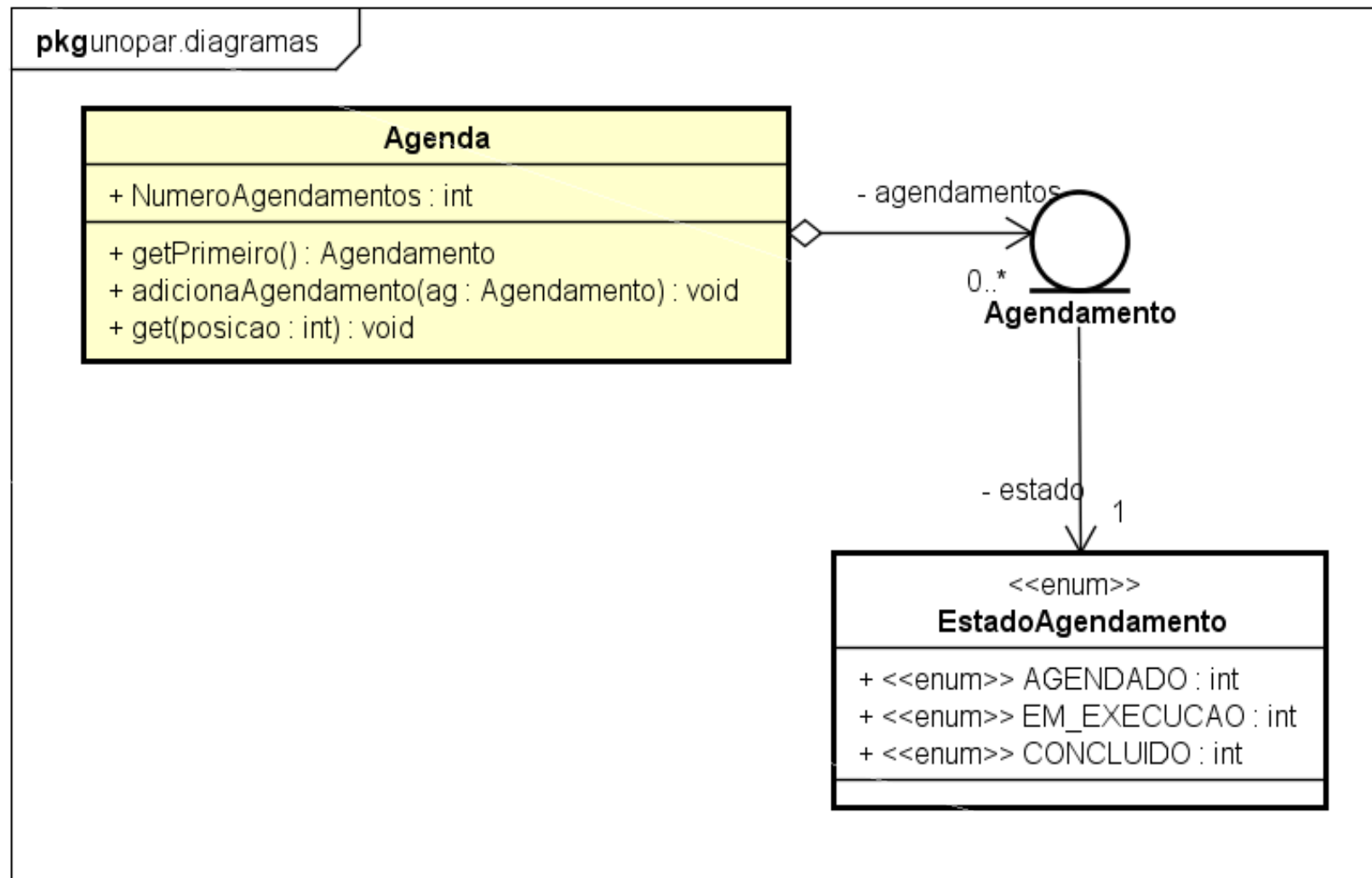
Qual estrutura de dados nos fornece estas características?



**NENHUMA!**

- Estruturas de dados NÃO TEM regra de negócios
- O programador deve utilizar estruturas de dados para implementar suas próprias classes adequadas ao domínio da aplicação

# API



```
public class Agenda {
    public Agenda() {}

    public int NumeroDeAgendamentos {
        get {
            return listaAgendamentos.Count;
        }
    }

    private List<Agendamento> listaAgendamentos = new List<Agendamento>();

    public void adicionaAgendamento(Agendamento agendamentoServico) {
        int insertionPos = 0;
        // encontra a melhor posição de inserção, onde elementoInserido.horario < prox.horario
        for(insertionPos = 0; insertionPos < listaAgendamentos.Count; insertionPos++) {
            Agendamento currentPosAgendamento = listaAgendamentos.ElementAt(insertionPos);

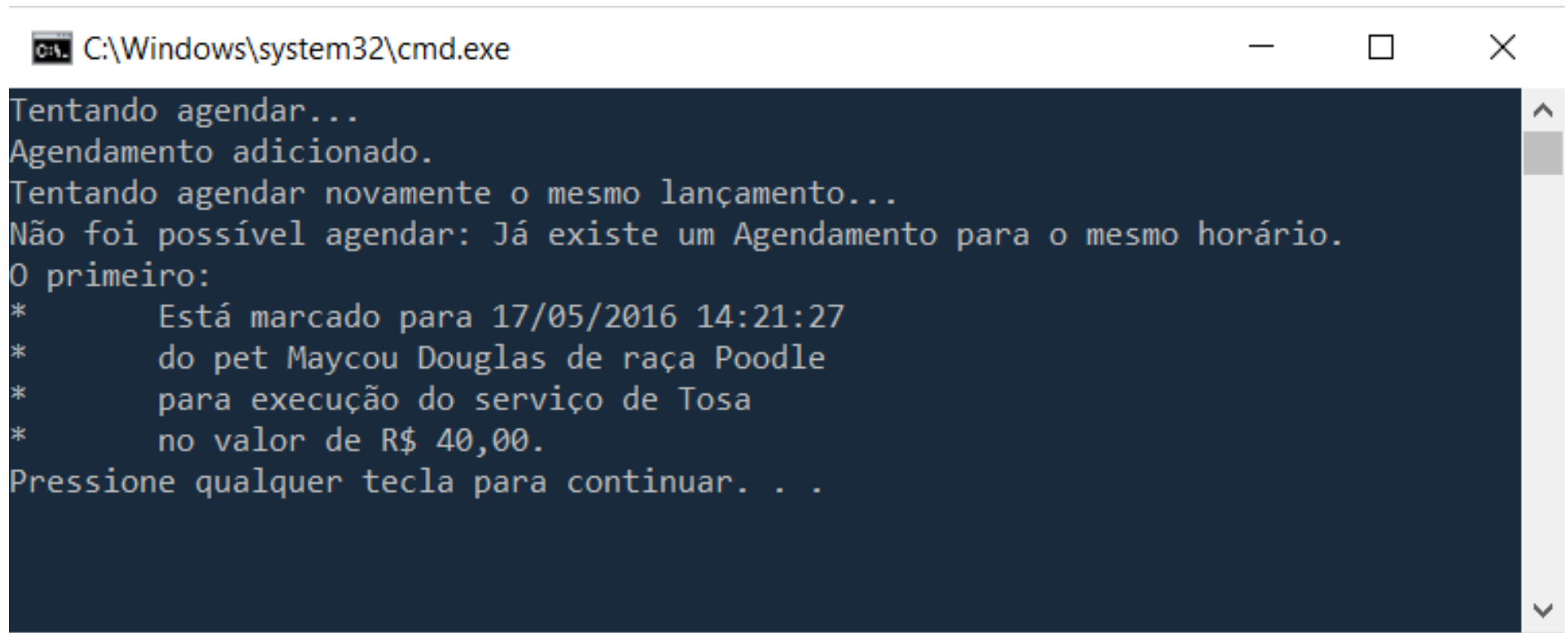
            // não podemos ter agendamento para o mesmo horário
            // TODO: Devemos checar por uma faixa de horário, considerando o tempo de serviço.
            if (agendamentoServico.HoraAgendamento == currentPosAgendamento.HoraAgendamento) {
                throw new HorárioAgendamentoException("Já existe um Agendamento para o mesmo horário.");
            }

            // se encontramos uma posição que o próximo agendamento é
            // "mais tarde" que o atual, é aqui que devemos inserir.
            if (agendamentoServico.HoraAgendamento < currentPosAgendamento.HoraAgendamento) {
                break;
            }
        }
        listaAgendamentos.Insert(insertionPos, agendamentoServico);
    }

    public Agendamento getPrimeiro() {
        return listaAgendamentos.FirstOrDefault();
    }

    public Agendamento get(int index) {
        return listaAgendamentos.ElementAt(index);
    }
}
```

# Saída



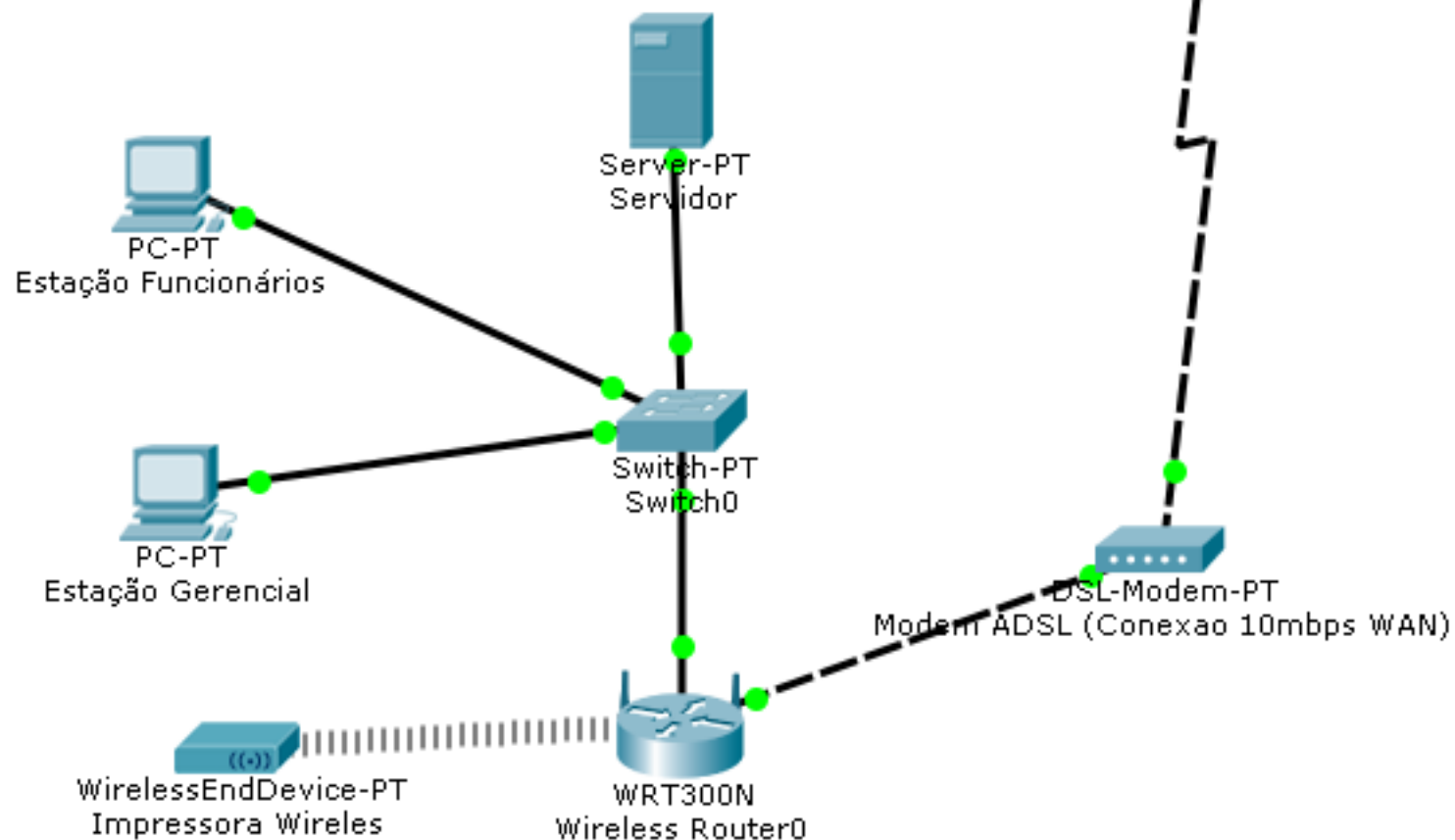
A screenshot of a Windows Command Prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe' and standard window controls. The command prompt has a dark blue background with white text. The output of a program is displayed, showing a scheduling process that fails due to a duplicate entry and then lists the details of the first entry.

```
C:\Windows\system32\cmd.exe
Tentando agendar...
Agendamento adicionado.
Tentando agendar novamente o mesmo lançamento...
Não foi possível agendar: Já existe um Agendamento para o mesmo horário.
O primeiro:
*      Está marcado para 17/05/2016 14:21:27
*      do pet Maycou Douglas de raça Poodle
*      para execução do serviço de Tosa
*      no valor de R$ 40,00.
Pressione qualquer tecla para continuar. . .
```

# Hardware e Rede

- Servidor físico
- Dispositivos móveis e “tradicionais”
- Acesso externo

## Externo



# Chegamos ao `</body>` :(

Toda documentação do trabalho está disponível  
(sob a GNU GPL v3) em:

<https://git.io/vot4g>

