Atividade pratica 3

Itamar Soldá Junior - 1992821

1. Introdução

- **Objetivo do Sistema**: Este sistema visa permitir o cadastro e gerenciamento de tutores e animais, com operações básicas de CRUD (Create, Read, Update, Delete) para ambos. Ele foi desenvolvido usando o framework **CherryPy** e implementa uma API REST para comunicação com o cliente.
- **Tecnologias Utilizadas**:
  - **CherryPy**: Framework para criação do servidor web.
  - **JSON**: Formato de troca de dados entre o cliente e o servidor.
  - **cURL**: Ferramenta usada para realizar os testes da API.

2. Estrutura do Sistema

O sistema foi estruturado com duas principais entidades:

- **Tutor**: Representa a pessoa responsável por um ou mais animais.
- **Animal**: Representa o animal cadastrado, com informações sobre nome, idade, sexo, peso, tamanho e o tutor responsável.

Classes e Métodos:

1. **Classe `TutorService`**:
   - **Métodos**:
     - `inserir`: Cadastra um novo tutor.
     - `buscar`: Lista todos os tutores ou busca um tutor específico pelo ID.
     - `atualizar`: Atualiza os dados de um tutor existente.
     - `deletar`: Exclui um tutor pelo ID.
2. **Classe `AnimalService`**:
   - **Métodos**:
     - `inserir`: Cadastra um novo animal, vinculado a um tutor.
     - `buscar`: Lista todos os animais ou busca um animal específico pelo ID.
     - `atualizar`: Atualiza os dados de um animal existente.
     - `deletar`: Exclui um animal pelo ID.

3. Funcionalidades Implementadas

- **Cadastro de Tutor**: Um tutor pode ser cadastrado com os campos **nome**, **telefone** e **endereco**.
- **Cadastro de Animal**: Um animal pode ser cadastrado com os campos **nome**, **idade**, **sexo**, **peso**, **tamanho**, e **tutor_id** (ID do tutor responsável).
- **Leitura de Tutor e Animal**: Através do método **GET**, é possível listar todos os tutores ou animais, ou então buscar um tutor ou animal específico utilizando seu ID.

- **Atualização de Tutor e Animal**: O método **PUT** permite a atualização dos dados de um tutor ou animal já existente.
- **Exclusão de Tutor e Animal**: O método **DELETE** permite excluir um tutor ou animal pelo seu ID.

4. API Endpoints

Aqui estão os principais **endpoints da API** para as operações de **tutores** e **animais**:

Método	Endpoint	Descrição
GET	`/tutores`	Lista todos os tutores.
GET	`/tutores/:id`	Busca um tutor específico pelo ID.
POST	`/tutores`	Cadastra um novo tutor.
PUT	`/tutores/:id`	Atualiza os dados de um tutor.
DELETE	`/tutores/:id`	Exclui um tutor pelo ID.
GET	`/animais`	Lista todos os animais.
GET	`/animais/:id`	Busca um animal específico pelo ID.
POST	`/animais`	Cadastra um novo animal.
PUT	`/animais/:id`	Atualiza os dados de um animal.
DELETE	`/animais/:id`	Exclui um animal pelo ID.

5. Processo de Testes

Os testes foram realizados utilizando **`cURL`**. Durante os testes, foram validadas todas as operações CRUD para **tutores** e **animais**, verificando que:

1. A criação de tutores e animais ocorre corretamente.
2. A leitura de tutores e animais (tanto a lista quanto a busca por ID) retorna os dados corretos.
3. A atualização de dados para tutores e animais funciona conforme esperado.
4. A exclusão de tutores e animais também está operando corretamente.

6. Resultados dos Testes

Aqui estão os resultados esperados para alguns testes básicos:

- **Cadastrar um Tutor**:
  - **Entrada**: Dados do tutor (nome, telefone, endereço).
  - **Saída Esperada**: JSON com a confirmação de cadastro e o ID do tutor.
- **Listar Tutores**:
  - **Entrada**: Requisição GET para `/tutores`.
  - **Saída Esperada**: JSON com uma lista de tutores cadastrados.
- **Cadastrar um Animal**:
  - **Entrada**: Dados do animal (nome, idade, sexo, peso, tamanho, tutor_id).
  - **Saída Esperada**: JSON com a confirmação de cadastro e o ID do animal.
- **Excluir um Animal**:
  - **Entrada**: Requisição DELETE para `/animais/:id`.

- **Saída Esperada**: JSON com a confirmação de exclusão.