Aplicația Teoriei și Practicii în Dezvoltarea Backend: O Abordare Integrată a Fundamentelor C#, OOP și MS SQL.

Elaborat

Chirita Stanislav

INTRODUCERE

Importanța temei se manifestă în modul în care aceasta contribuie la evoluția și eficiența dezvoltării backend în domeniul software. Prin abordarea integrată a fundamentelor C#, OOP și MS SQL, se dezvăluie o perspectivă esențială asupra procesului de dezvoltare a aplicațiilor backend. Această abordare nu se limitează doar la teorie, ci aduce în prim-plan practica și aplicarea concretă a cunoștințelor.

Prin înțelegerea profundă a limbajului C#, dezvoltatorii pot construi aplicații mai consistente și mai ușor de întreținut. De asemenea, integrarea principiilor OOP permite gestionarea mai eficientă a complexității în dezvoltarea software, ceea ce duce la aplicații mai robuste și scalabile. Gestionarea bazelor de date MS SQL este esențială în asigurarea stocării și manipulării corecte a datelor, având un impact semnificativ asupra performanței și securității aplicațiilor.

Descrierea tehnologiilor utilizate

C# (C-Sharp)

Este un limbaj de programare modern, dezvoltat de Microsoft, utilizat pentru dezvoltarea aplicațiilor pe platforma .NET. C# oferă o sintaxă clară și robustă, suport pentru programarea orientată pe obiect și facilitează dezvoltarea aplicațiilor scalabile și securizate.

Programarea Orientata pe Obiect (OOP)

Este o paradigmă de programare care se concentrează pe organizarea codului în jurul obiectelor și a interacțiunilor între acestea. OOP facilitează dezvoltarea modulară și reutilizarea codului.

Microsoft SQL(MS SQL)

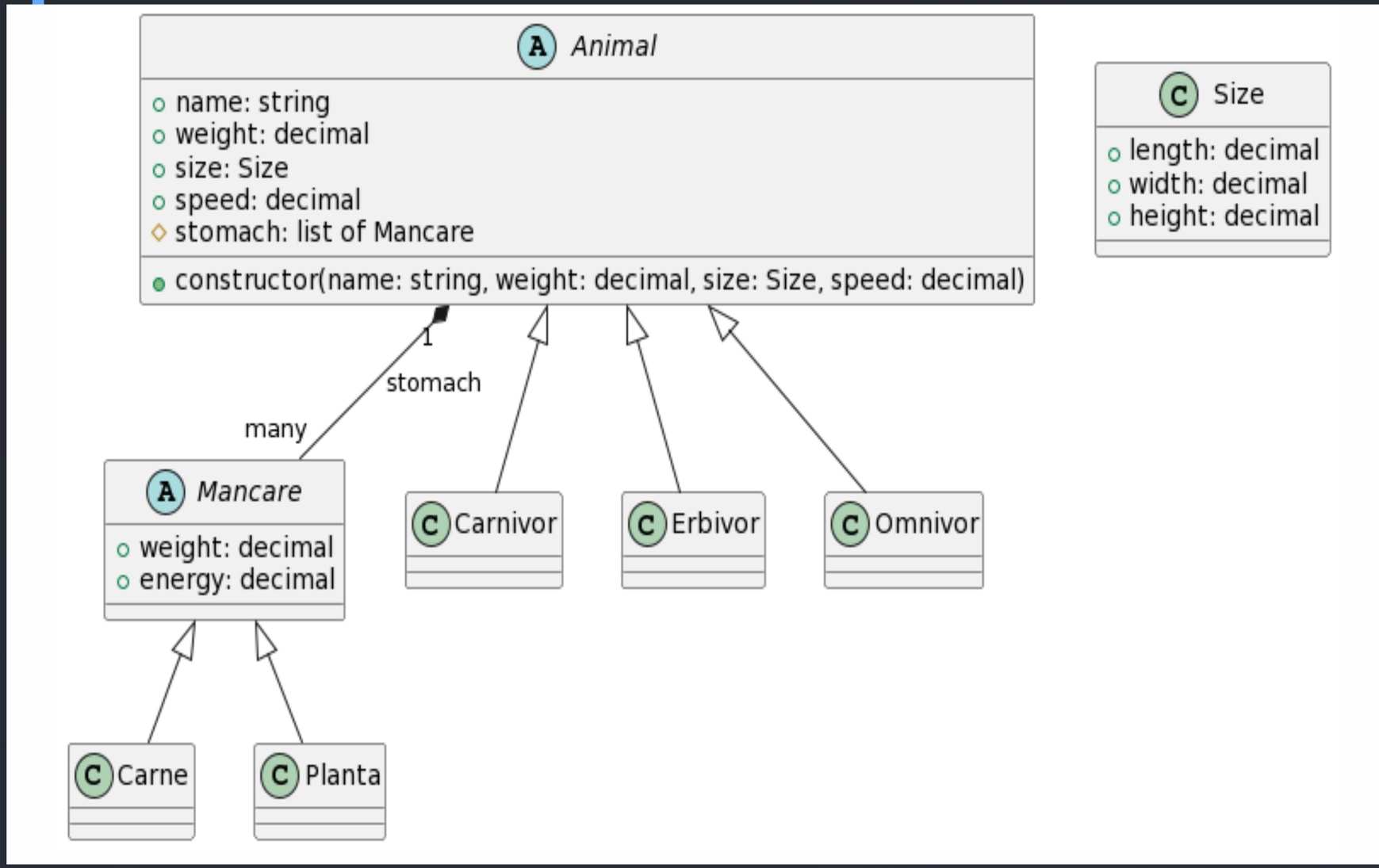
Este un sistem de management al bazelor de date relaționale dezvoltat de Microsoft. MS SQL oferă funcționalități avansate pentru stocarea și gestionarea datelor, inclusiv suport pentru tranzacții, securitate și optimizare a interogărilor.

Modelarea și proiectarea

Sarcina 4 implică crearea unei ierarhii de clase pentru a gestiona animalele şi hrana lor. Mai întâi, a fost creată o clasă de bază numită "Animal" cu proprietăţile "Nume" şi "Energie". Apoi, au fost create două clase derivate, "Carnivoră" şi "Erbivor", fiecare cu proprietăţile lor specifice ("TipCarne" pentru carnivore şi "TipPlanta" pentru erbivore).

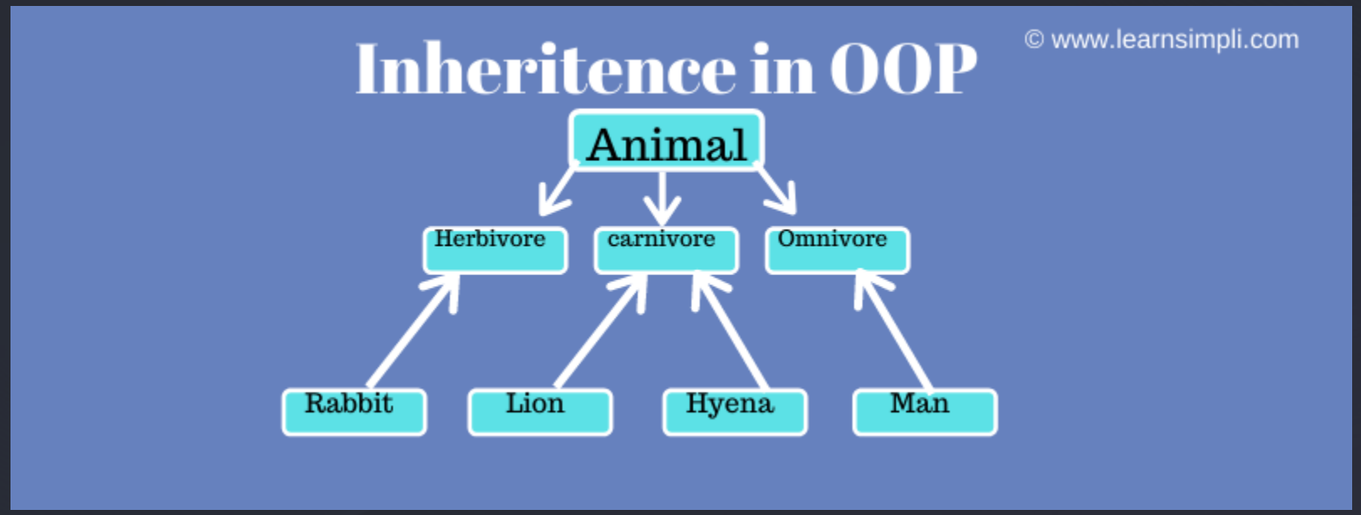
În plus, a fost creată o clasă separată numită "Mâncare" cu proprietăţile "Nume" şi "Energie" pentru a gestiona informaţii despre alimente.

Pentru a testa funcţionalitatea acestor clase, a fost creată o clasă de testare numită "Test", care include metode pentru crearea, modificarea şi utilizarea obiectelor de tip "Animal", "Carnivoră", "Erbivor" şi "Mâncare". Aceasta oferă o structură de bază pentru lucrul cu animale şi hrana lor în cadrul programului.



Crearea unei clase de bază pentru toate animalele

```cs
using System;
using System.Runtime.InteropServices;

namespace T1
{

    public struct dimensiune
    {
        public decimal lungime { get; set; }
        public decimal latime { get; set; }
        public decimal inaltime { get; set; }
        public dimensiune (decimal _lungime, decimal _latime, decimal _inaltime)
        {
            lungime = _lungime;
            latime = _latime;
            inaltime = _inaltime;
        }
    };
    public abstract class Animal
    {
    }
    public class Carnivor : Animal
    {
    }
    public class Erbivor : Animal
    {
    }
    public class Omnivor : Animal
    {
    }

    public abstract class Mancare
    {
    }
    public class Carne : Mancare
    {

    }
    public class Planta : Mancare
    {

    }
    public class Program
    {
    }
}
```



Sarcina 5

Din programul de practică "Your first program architecture" durează 10 zile și constă în crearea unui proiect pentru proiectarea unei ierarhii de clase ce alcătuiesc un Espressor. Candidații trebuie să utilizeze toate conceptele de Programare Orientată pe Obiect (POO) cunoscute pentru a crea această ierarhie, incluzând o clasă principală numită Espressor și diverse componente. Proiectul trebuie să includă metode publice user-friendly pentru interacțiunea cu sistemul, iar structura sistemului trebuie să fie cât mai apropiată de limbajul natural și de structura unui espressor fizic. În punctul de intrare al aplicației, trebuie să se instanțieze Espressorul și să se expună opțiuni de interacțiune la consolă pentru utilizator.

Diagrama UML (Espressor)

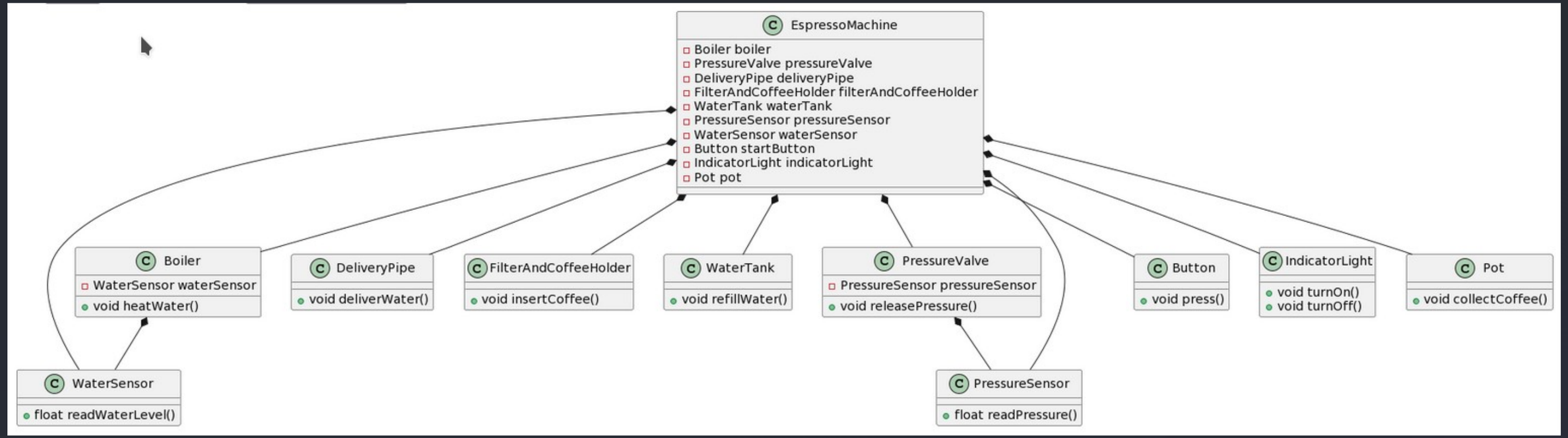


Diagrama UML prezentată ilustrează arhitectura unei mașini de espresso, evidențiind clasele și relațiile dintre acestea. Clasa centrală, `EspressoMachine`, conține multiple instanțe ale altor clase care reprezintă diverse componente ale espressorului. De exemplu, `EspressoMachine` are un `WaterTank`, un `Boiler`, un `PressureValve`, și așa mai departe. Fiecare clasă componente are metode specifice, precum `refillWater()` pentru `WaterTank`, sau `heatWater()` pentru `Boiler`. Relațiile sunt reprezentate prin linii care conectează `EspressoMachine` cu fiecare componentă, semnificând că `EspressoMachine` conține sau folosește aceste componente în procesul de preparare a cafelei. Diagrama servește drept un ghid pentru structura de cod și interacțiunile dintre obiectele dintr-un program de simulare a funcționării unui espressor de cafea.

Codul

Codul prezentat descrie funcționarea unui aparat de făcut espresso, folosind principiile programării orientate pe obiect. Fiecare componentă a espressorului este reprezentată printr-o clasă distinctă, cum ar fi `WaterTank`, `Boiler`, `PressureValve`, etc. Aceste clase conțin metode care simulează diferite acțiuni ale componentelor reale, cum ar fi umplerea rezervorului cu apă, încălzirea apei și eliberarea presiunii. Clasa principală, `EspressoMachine`, integrează toate aceste componente și oferă o metodă `MakeCoffee` pentru a simula procesul de preparare a cafelei. Metoda `MakeCoffee` coordonează acțiunile diferitelor componente, verificând senzorii și controlând luminile indicatoare pentru a finaliza procesul de preparare a cafelei. Programul este testat în metoda `Main` a clasei `Program`, unde se creează o instanță a `EspressoMachine` și se apelează metoda `MakeCoffee`.

```
class WaterTank {…

2 references
class Boiler {…

2 references
class PressureValve {…

2 references
class DeliveryPipe {…

2 references
class FilterAndCoffeeHolder {…

2 references
class PressureSensor {…

2 references
class WaterSensor {…

2 references
class Button {…

2 references
class IndicatorLight {…

2 references
class EspressoMachine {…

0 references
class Program {…
```

Concluzii

În concluzie, abordarea integrată a fundamentelor C#, programării orientate pe obiect (OOP) și utilizării bazei de date Microsoft SQL (MS SQL) în dezvoltarea backend reprezintă o strategie eficientă pentru crearea aplicațiilor backend robuste și scalabile. Această abordare oferă dezvoltatorilor un set de instrumente și cunoștințe necesare pentru a proiecta, dezvolta și întreține soluții software de înaltă calitate.

Prin combinarea puterii limbajului C# pentru implementarea logicii de afaceri, principiilor OOP pentru structurarea și modularizarea codului și MS SQL pentru gestionarea datelor, dezvoltatorii pot crea aplicații care îndeplinesc cerințele complexe ale industriei IT actuală. Această abordare permite dezvoltatorilor să creeze aplicații eficiente, ușor de întreținut și scalabile, esențiale pentru satisfacerea nevoilor organizațiilor și a utilizatorilor finali.