

FOODMANAGER

aplicație universală pentru
gestionarea comenziilor în unități de
alimentație publică

A efectuat:

st. gr. TI-224, Russu Iulia

A verificat:

Bîtca Ernest



INTRODUCERE

Acest proiect se concentrează pe implementarea unei aplicații pentru gestionarea comenzi în unități de alimentație publică, de exemplu - o cafenea.

Sunt integrate cinci șabloane de proiectare: **Factory**, **Singleton**, **Decorator**, **Observer** și **Strategy**.

Prin acest exemplu practic, se evidențiază rolul fiecărui șablon în soluționarea unor aspecte specifice ale aplicației, precum crearea obiectelor, gestionarea instanțelor unice, extinderea dinamică a funcționalităților.



PROIECTAREA SISTEMULUI

DIAGRAMA USE-CASE

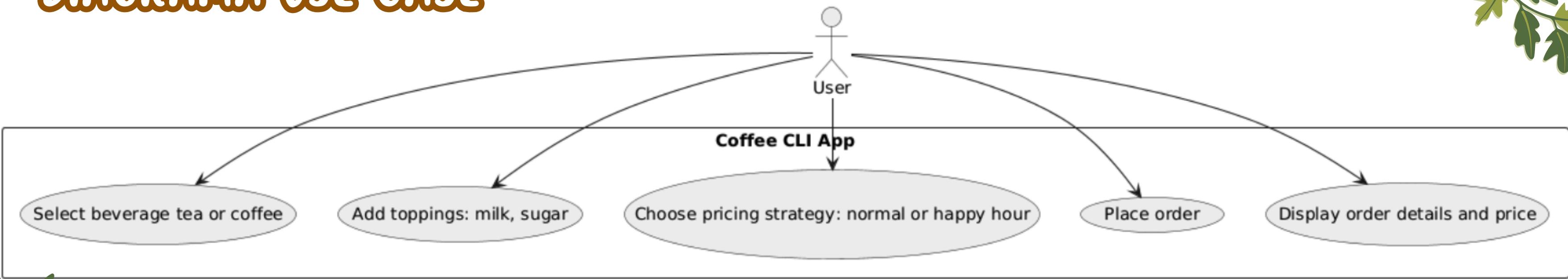


DIAGRAMA SECVENTIALĂ

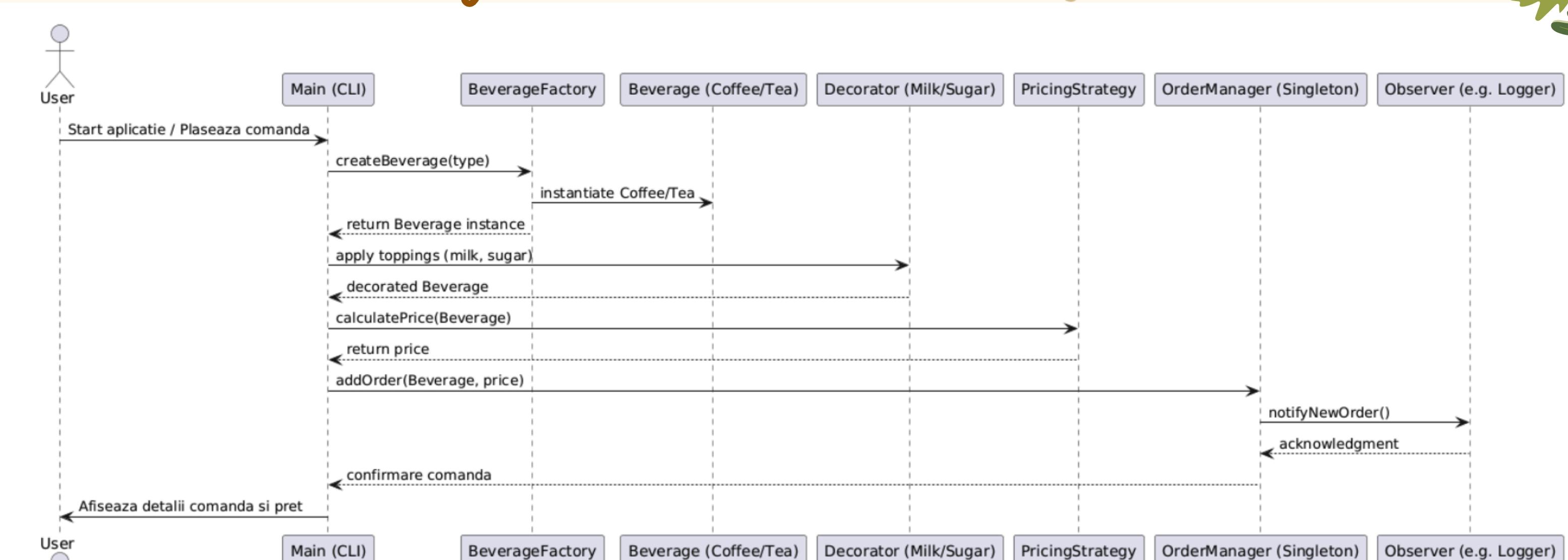


DIAGRAMA DE CLASE

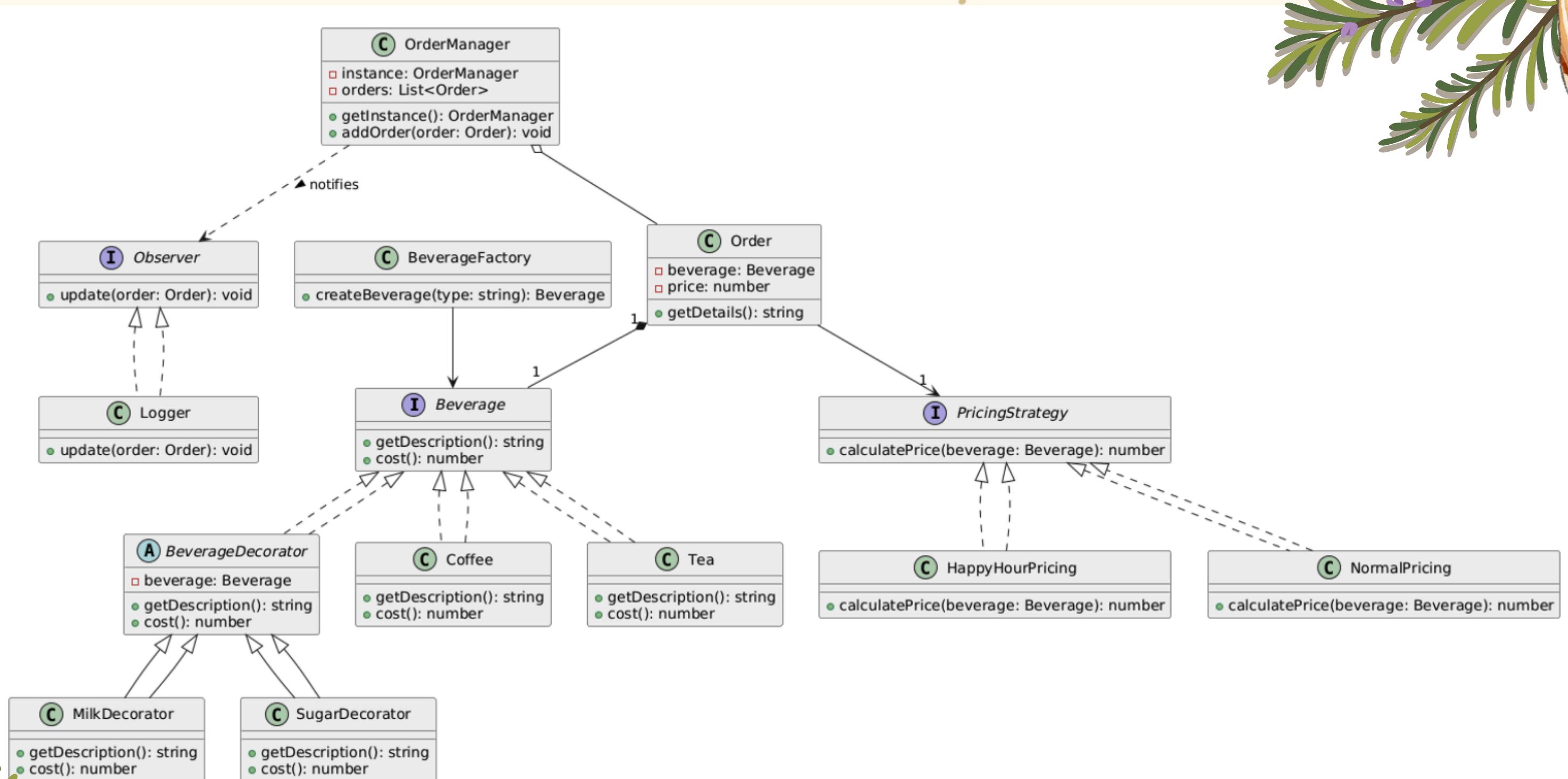
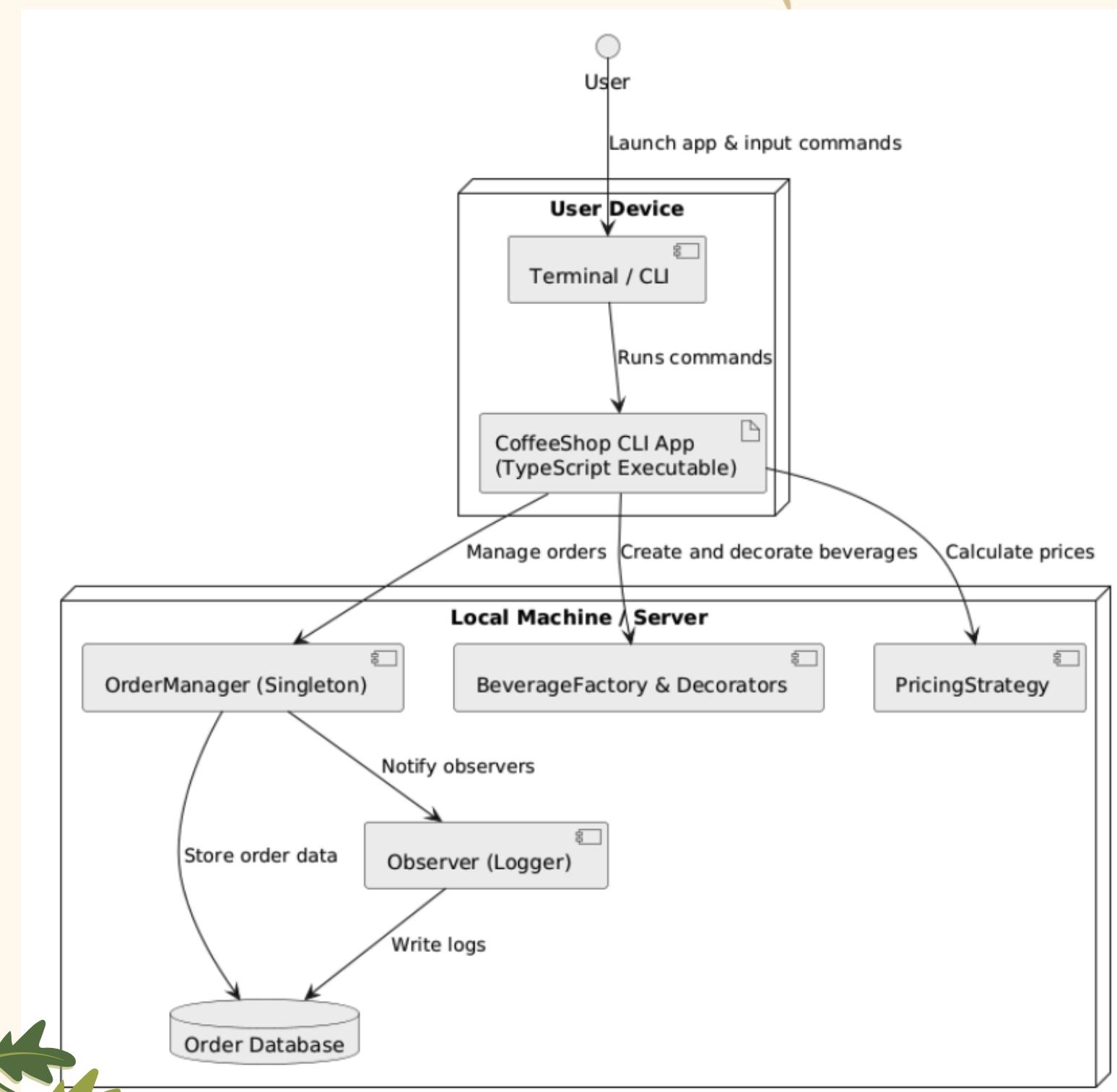
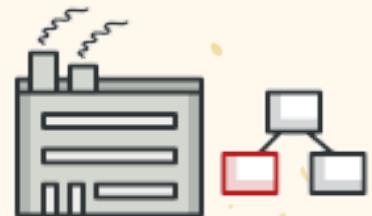


DIAGRAMA DE TIP DEPLOYMENT



IMPLEMENTAREA

TS



FACTORY PATTERN

clasa: BeverageFactory

rol: primește un tip de băutură ca string și returnează o instanță a clasei respective (Coffee, Tea, etc.).

```
case "coffee":  
    return new Coffee();  
case "tea":  
    return new Tea();
```

*creațional



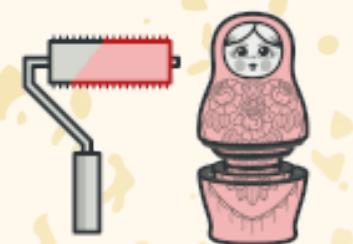
SINGLETON PATTERN

clasa: OrderManager

rol: sigură că există o singură instanță a managerului pe toată durata de rulare a aplicației.

```
static getInstance(): OrderManager {  
    if (!OrderManager.instance) {  
        OrderManager.instance = new OrderManager();  
    }  
    return OrderManager.instance;  
}
```

*creațional



DECORATOR PATTERN

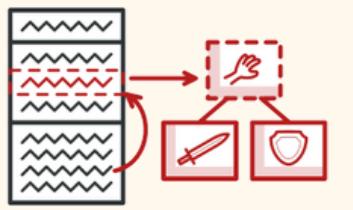
clasa: MilkDecorator

rol: adăuga funcționalitate suplimentară fără a modifica clasele existente

```
getDescription(): string {  
    return this.beverage.getDescription()  
    + ", lapte";  
}  
getCost(): number {  
    return this.beverage.getCost() + 2;  
}
```

*structural





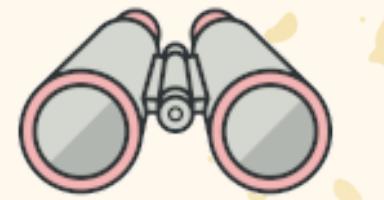
STRATEGY PATTERN

clasa: NormalPricing

rol: aplică dinamic modificări în funcție de context, fără a modifica codul client.

```
export class HappyHourPricing implements  
PricingStrategy {  
calculatePrice(beverage: Beverage): number {  
return beverage.cost() * 0.8; // 20%  
}  
}
```

*de comportament



OBSERVER PATTERN

clasa: MilkDecorator

rol: decouplează emiterea evenimentelor de componentele care reacționează la ele.

```
update(order: Order): void {  
console.log("[LOG] Comanda  
înregistrată: ", order.getDetails());  
}
```

*de comportament



INTERFAȚA APlicațIEI

MENIU

```
PS C:\Users\russu\OneDrive\Desktop\UTM III\TMPP\Individual\src> node index.js
== BUN VENIT LA CAFENEA ==
Alege bautura (1 - Cafea, 2 - Ceai): 2
Vrei sa adaugi lapte? [y/n]: y
Vrei sa adaugi zahar? [y/n]: n
Alege tipul de pret (1 - Normal, 2 - Happy Hour): 2
Comanda: Ceai, lapte | Total: 8.00 lei
Comandă nouă: Ceai, lapte - 10 lei
Vrei sa mai comanzi? [y/n]:
```

PRODUSE DISPONIBILE

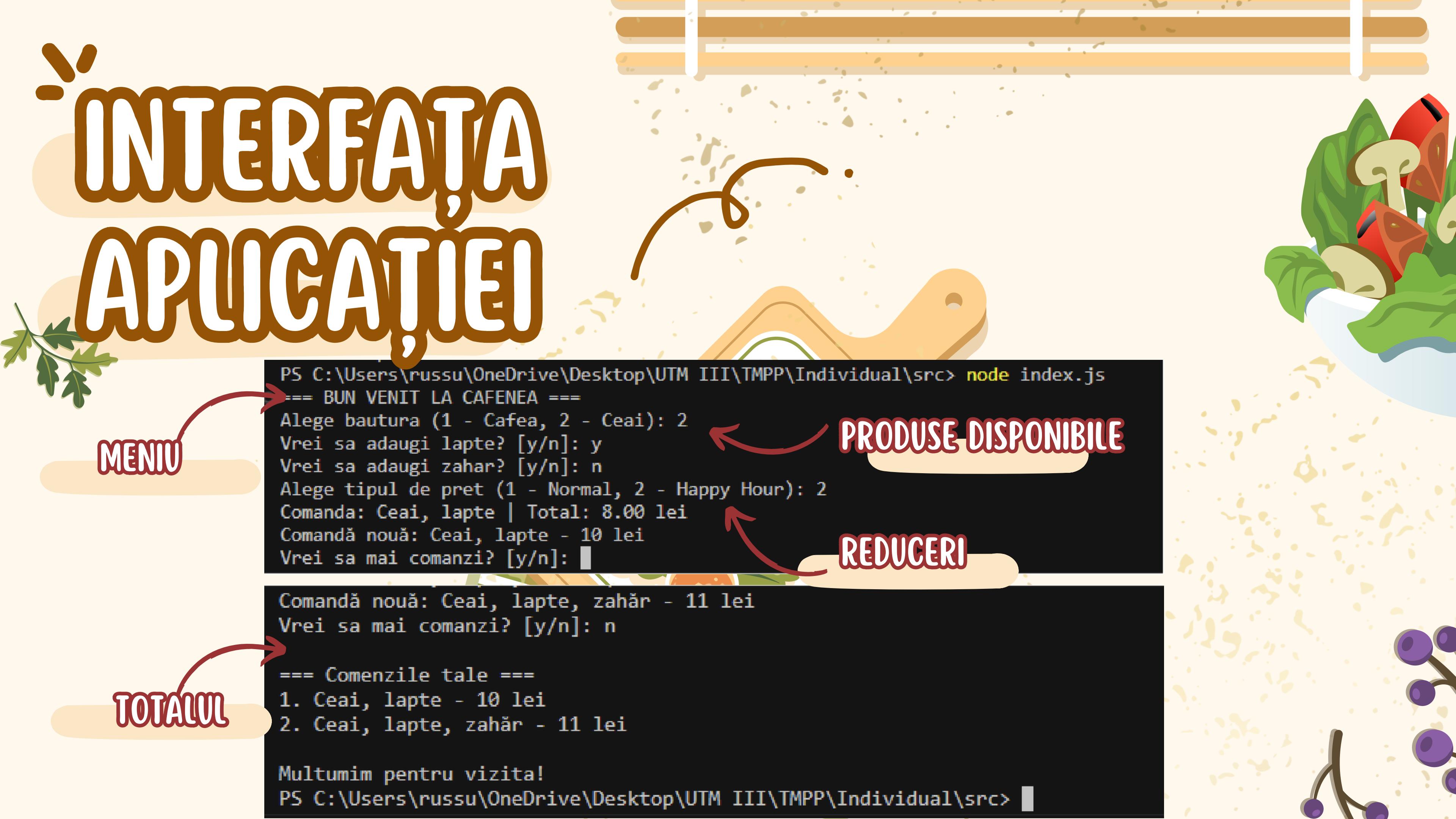
REDUCERI

TOTALUL

```
Comandă nouă: Ceai, lapte, zahăr - 11 lei
Vrei sa mai comanzi? [y/n]: n

== Comenzile tale ==
1. Ceai, lapte - 10 lei
2. Ceai, lapte, zahăr - 11 lei

Multumim pentru vizita!
PS C:\Users\russu\OneDrive\Desktop\UTM III\TMPP\Individual\src>
```



PERSPECTIVA APLICAȚIEI

Aplicația oferă o soluție modernă și eficientă pentru casieri, permitând calculul rapid al costului total al produselor și adăugarea ușoară a toppingurilor personalizate, precum zahăr în ceai sau ingrediente suplimentare pe pizza, la fel și aplicarea reducerilor diferite.

Datorită versatilității sale, aplicația se adaptează cu ușurință oricărui tip de instituție alimentară publică – de la cafenele și pizzerii până la cantine sau restaurante – simplificând procesul de vânzare și îmbunătățind experiența clientului.

Un exemplu actual ar fi utilizarea aplicației în cadrul cantinei UTM, pentru a ușura calculul manual al prețului fiecărei comenzi a studenților, care deseori este greu de calculat, introducând anterior reduceri de ziua studentului sau alte sărbători.



CONCLUZIE

Proiectul realizat demonstrează importanța și utilitatea șabloanelor de proiectare în dezvoltarea aplicațiilor software moderne.

Prin aplicarea practică a cinci șabloane - **Factory**, **Singleton**, **Decorator**, **Strategy** și **Observer** - aplicația a fost construită într-un mod modular, extensibil și ușor de întreținut.