

**PROIECT SISTEME MULTIAGENT**

**Conducător științific:**

**Conf. Dr. Ing. Dan ROSENBERG**

**Studenți:**

**Diana DUMITRU**

**Bogdan PATRANIA**

**Marina RADU**

**Gabriel TOMUȚA**

**BRAȘOV, 2024**

# Tema proiectului

## Tema Propusă

Tema de proiect A: (9)

Realizati o aplicație JADE care sa creeze 5 agenți care să primească mesaje de la o interfață grafică. Exemplificați folosind agenți BDI.

## Tema Aleasă

Restaurant Inteligent

Agenți:

ManagerAgent: Gestionează operațiunile generale ale restaurantului, atribuie sarcini altor agenți și se ocupă de solicitările utilizatorilor. (Primește mesaje de la utilizator)

ChefAgent: Primește comenzi de la ManagerAgent, pregătește mâncarea și informează ManagerAgent când este gata.

WaiterAgent: Primește comenzi de la ManagerAgent, livrează mâncarea clienților și ia noi comenzi de la utilizator.

TableAgent: Reprezintă o masă în restaurant, ține evidența ocupării și a comenzilor.

KitchenDisplayAgent: Afișează comenzile actuale de mâncare pentru ChefAgent.

Interacțiunea Utilizatorului: Utilizatorul poate interacționa cu ManagerAgent prin mesaje text (simulate) pentru a:

Face o rezervare pentru o masă.

Comandă mâncare din meniu.

# Implementarea soluției

Proiectul nostru utilizează JADE (Java Agent Development Framework) pentru a simula managementul unui restaurant printr-o arhitectură multi-agent. Scopul acestui proiect este de a demonstra cum agenții software pot interacționa pentru a îndeplini sarcini complexe într-un mediu dinamic. Sistemul nostru folosește agenți de tip BDI (Belief-Desire-Intention), un model care permite agenților să ia decizii bazate pe convingeri, dorințe și intenții, ceea ce îi face capabili să gestioneze situații complexe și imprevizibile.

Modelul BDI este inspirat din teoria acțiunii practice și permite agenților să funcționeze într-un mod similar cu cel al oamenilor. Agenții noștri BDI sunt programați să perceapă mediul înconjurător, să actualizeze starea internă a convingerilor lor (beliefs), să genereze dorințe (desires) pe baza obiectivelor de atins și să formuleze intenții (intentions) care ghidează acțiunile lor pentru a îndeplini aceste obiective. Proiectul simulează un mediu realist de operare a unui restaurant. Acest mediu este guvernat de cei cinci agenți, responsabili de diferite acțiuni create în baza unor scenarii definite apriori.

Agentul Manager deține responsabilitatea de bază. Clientul, în acest caz, utilizatorul, se adresează strict managerului pentru îndeplinirea diverselor cereri (rezervarea mesei, comandarea unui tip de mâncare, etc.) pe care agentul Manager le trimite mai departe, prin inițierea unei comunicații cu restul agenților, în funcție de tipologia cerinței.

Agentul Table este responsabil de mesele restaurantului. Acesta deține informații precum statusul mesei (ocupată, liberă), stadiul clientului (mănâncă, a terminat de mâncat) cât și tipul de mâncare comandat la o anumită masă, împreună cu prețul acestuia. Toate informațiile sunt actualizate în timp real în baza datelor primite de la ceilalți agenți. Comunicarea agentului se realizează în prealabil cu agentul Waiter în cazul unei comenzi, cu agentul Checkout în cazul inițierii procesului de plată și cu agentul Manager în cazul rezervării unei mese sau în cazul sanitizării acesteia.

Agentul Waiter simulează rolul unui chelner. Acesta are ca scop doar transportul comenzii clientului către agentul Chef, furnizarea produsului culinar și mai apoi curățarea mesei.

Agentul Chef reprezintă un agent cu scopuri reduse, singurul său obiectiv fiind pregătirea preparatelor culinare.

Agentul Checkout este un agent de tip casier. Acesta reține costul intern suportat de restaurant pentru obținerea diverselor tipuri de mâncare și prețul plătit de către consumator. În momentul închiderii restaurantului, acesta returnează profitul obținut.

Fluxul de interacțiuni dintre acești agenți poate fi descris prin intermediul unor scenarii. Spre exemplu în cazul în care utilizatorul dorește sa rezerve o masă, acesta trimite o solicitare către agentul Manager. Agentul Manager verifică disponibilitatea meselor prin comunicarea cu agentul Tabel. Dacă există mese libere, agentul Manager alocă una dintre acestea clientului și confirmă rezervarea. În caz contrar, informează clientul că toate mesele sunt ocupate.

În cazul procesului de comandare a mâncării, clientul își exprimă dorința de a comanda un anumit preparat prin intermediul agentului Manager. Managerul transmite această comandă agentului Tabel, care înregistrează detaliile comenzii și actualizează suma totală de plată. Atunci când clientul dorește să modifice comanda, agentul Manager comunică schimbările agentului Tabel pentru a o actualiza. Odată ce clientul finalizează plasarea comenzii, agentul Manager transmite instrucțiunile către agentul Waiter, care mai apoi transmite cererea agentului Chef pentru prepararea mâncării. După ce mâncarea este pregătită, agentul Waiter o livrează la masa clientului și actualizează Agentul Tabel cu detaliile livrării

Când clientul dorește să achite nota de plată, acesta trimite o solicitare către agentul Manager. Managerul comunică cu agentul Tabel pentru a confirma totalul de plată. Apoi, agentul Tabel transmite suma încasată către agentul Checkout, în vederea calculării profitului total de la finalul zilei. Managerul instruiește agentul Waiter să curețe masa. După curățenie, agentul Waiter informează agentul Tabel că masa este liberă.

La finalul zilei, agentul Manager inițiază procedura de închidere a restaurantului. Managerul comunică cu agentul Tabel pentru a verifica starea meselor și pentru a se asigura că toate sunt libere. Apoi, managerul instruiește agentul Checkout să returneze profitul încasat. Odată ce toate aceste operațiuni sunt încheiate, restaurantul este considerat închis.

Scenariile enumerate reprezintă bază de la care atributele BDI ale fiecărui agent au fost create. Întreaga funcționalitate a sistemului a fost construită de-a lungul unei logici de program derivată din scenariile prezentate. Utilizarea acestui sistem pentru managementul restaurantului oferă numeroase avantaje, printre care flexibilitatea în gestionarea resurselor și scalabilitatea sistemului. Fiecare agent poate funcționa independent, dar poate colabora eficient cu ceilalți agenți, ceea ce permite sistemului să se adapteze rapid la schimbări sau probleme neașteptate. În plus, distribuirea sarcinilor între agenții specializați reduce timpul de răspuns și crește eficiența operațională. Acest proiect evidențiază potențialul tehnologiei multi-agent și a modelului BDI în diverse domenii, contribuind la dezvoltarea soluțiilor inovative în gestionarea resurselor și proceselor

# Programul

|  |
| --- |
| public class WaiterAgent extends RestaurantAgent {  // Beliefs  String location; // Current location  boolean has\_food; // Does it carry food  // Desires  LinkedList<String> desires = new LinkedList<>();  @Override  protected void setup() {  location = "table\_0"; // Initial location  has\_food = false;  addBehaviour(new CyclicBehaviour() {  @Override  public void action() {  if (!desires.isEmpty()) {  String current\_desire = desires.getFirst();  // Choose action  if (current\_desire.startsWith("location")) {  walk(current\_desire.split(":")[1]);  desires.removeFirst();  }  else ...  }  else {  this.block();  }  }  });  }  protected void interpretMessage(String msg) {  if (msg.startsWith("Send order")) {  desires.add(String.format("order:%s", msg.substring(11)));  }  else ...  this.doWake();  }  public void walk(String target) {  if (!location.equals(target)) {  sleep(500); // Simulate walking  location = target;  }  }  public boolean sendOrderToChef(String order) {...}  public boolean deliverOrder(int table\_num) {...}  public boolean cleanTable(int table\_num) {...}  } |

# Bibliografie

[1] [https://download.actoron.com/docs/releases/jadex-3.0.0-RC51/jadex- mkdocs/BDI%20V3%20Tutorial/01%20Introduction/](https://download.actoron.com/docs/releases/jadex-3.0.0-RC51/jadex-%20%20mkdocs/BDI%20V3%20Tutorial/01%20Introduction/)

[2] <https://chatgpt.com/>

[3] Floroian, D. (2023). Sisteme multiagent. Alba Iulia: Editura Albastră.