

POLITECNICO DI TORINO

*Laurea di 1° livello in Ingegneria Gestionale
Classe L-8 Ingegneria dell'Informazione*



Gestione evento culinario in agriturismo

Relatore

Prof. Fulvio Corno

Candidato

Davide Visconti

Anno Accademico
2018/2019

Indice

1. PROPOSTA DI PROGETTO	3
1.1 Studente proponente.....	3
1.2 Titolo della proposta	3
1.3 Descrizione del problema proposto	3
1.4 Descrizione della rilevanza gestionale del problema.....	3
1.5 Descrizione dei data-set utilizzati per la valutazione.....	3
1.6 Descrizione preliminare degli algoritmi coinvolti	4
1.7 Descrizione funzionalità applicazione software	4
2. DESCRIZIONE DEL PROBLEMA AFFRONTATO	4
3. DESCRIZIONE DEI DATA-SET UTILIZZATI	6
4. DESCRIZIONE DELLE STRUTTURE DATI E DEGLI ALGORITMI UTILIZZATI.....	7
5. DIAGRAMMA DELLE CLASSI	13
6. VIDEATE DELL'APPLICAZIONE E RISULTATI SPERIMENTALI	14
7. VALUTAZIONE DEI RISULTATI E CONCLUSIONI	16

1. PROPOSTA DI PROGETTO

1.1 Studente proponente

S227038 Davide Visconti

1.2 Titolo della proposta

Gestione e ottimizzazione di un evento culinario in agriturismo.

1.3 Descrizione del problema proposto

L'applicazione permette all'utente di scegliere una manifestazione all'aperto da svolgersi tra quelle presenti e di selezionare dei piatti e delle bevande che verranno serviti durante lo svolgimento dell'evento. Compito dell'applicazione sarà quello di simulare l'arrivo e gli ordini dei diversi clienti, tenendo conto di tutti i parametri utili a fornire un'adeguata valutazione economica dell'evento, per poi compararlo con altri e decidere quale sia la manifestazione migliore da svolgersi a seconda delle preferenze.

1.4 Descrizione della rilevanza gestionale del problema

Il problema è rilevante dal punto di vista gestionale perché aiuta l'utente a valutare la manifestazione, fornendogli indicatori che dovrebbero risultare molto utili nell'effettuare la scelta reale dell'evento da organizzare.

1.5 Descrizione dei data-set utilizzati per la valutazione

Il data-set utilizzato è stato creato da me e comprende tre tabelle: manifestazione, bevanda, piatto; i dati inseriti sono stati discussi con un mio amico che si occupa di contabilità in un agriturismo e che, successivamente, sarà il principale fruitore del programma.

1.6 Descrizione preliminare degli algoritmi coinvolti

L'algoritmo è principalmente un algoritmo di simulazione: esso simula lo svolgimento multiplo della manifestazione selezionata e stampa poi nell'interfaccia i parametri maggiormente significativi per la valutazione dell'evento.

1.7 Descrizione funzionalità applicazione software

L'applicazione permette all'utente di selezionare una manifestazione attraverso una ComboBox e diversi piatti e bevande da due menù a tendina; una volta selezionato il tutto, si devono inserire in apposite caselle di testo dei parametri utili alla simulazione e, schiacciando sul bottone "simula", si può lanciare il programma che si occuperà di stampare i parametri necessari.

2. DESCRIZIONE DEL PROBLEMA AFFRONTATO

Ho avuto l'idea di sviluppare questo programma per aiutare a pianificare l'organizzazione e ad ottimizzare (a livello costi e soddisfazione della clientela) un evento culinario in un agriturismo. Nel dettaglio, oltre alla creazione del database che verrà discusso in seguito, l'utente sceglie una manifestazione, dei piatti e delle bevande (i dettagli di queste classi verranno anch'essi descritti in seguito); dopo aver fatto ciò, si devono inserire nelle caselle di testo cinque differenti parametri:

- Il numero di dipendenti che si occuperà di prendere le ordinazioni;
- Il numero di dipendenti che sarà al servizio dei piatti (in tale numero vi è incorporato anche un dipendente che si occuperà continuamente di servire le bevande);
- Un tempo di riordino medio (in minuti, variabile stocasticamente del 40%) dopo il quale un cliente tornerà a mettersi in coda per effettuare un nuovo ordine;
- La paga oraria dei dipendenti (in euro);
- Il numero di simulazioni dello stesso evento da effettuarsi.

Una volta selezionati e inseriti gli opportuni valori, il programma procederà con la simulazione: a partire dalla manifestazione verranno generati un numero di clienti (variabile stocasticamente del 30% da una simulazione ad un'altra), ognuno con il proprio budget (variabile anch'esso stocasticamente tra le diverse simulazioni) e il loro arrivo verrà suddiviso in tre fasce da un'ora (per ogni evento l'inizio è previsto per le 19) con maggiore concentrazione di clienti nella seconda fascia oraria (20-21).

Una volta arrivato il cliente si metterà in coda per ordinare e dovrà attendere un certo tempo dovuto al numero di persone davanti a lui e al numero di dipendenti in cassa. Una volta fatta l'ordinazione, il cliente si metterà in attesa nella coda per il servizio; qui dovrà attendere un tempo proporzionale a quante altre persone ha davanti e a cosa hanno ordinato (i diversi piatti hanno un tempo di preparazione "istantaneo", inteso cioè come il tempo di preparazione del piatto considerando già pronti gli ingredienti necessari; facendo un esempio, per preparare un panino con il "pulled pork" si stima un tempo di 60 secondi, non viene tenuto in considerazione il fatto che la preparazione del "pulled pork" richiede circa 12 ore di cottura perché non fa parte dello scopo del programma pianificare la preparazione dei cibi, bensì si vuole fornire la stima della quantità necessaria).

Se il tempo di attesa in coda, in cassa o al servizio, supera i 30 minuti il cliente risulterà insoddisfatto; egli potrà risultare insoddisfatto di solo un'attesa oppure di entrambe. Una volta terminata la coda per il servizio riceverà il piatto e dopo il tempo stimato per il riordino tornerà in coda alla cassa.

Il programma terrà conto di tutti gli ordini effettuati dai vari clienti che possono comprendere un piatto e una bevanda (condizione obbligatoria al primo ordine che si effettua) oppure solo un piatto o solo una bevanda.

Alla fine della simulazione il software terrà conto del costo di organizzazione dell'evento, del guadagno, dell'incasso, di tutte le quantità di piatti e bevande ordinate, del tempo di attesa medio e massimo in cassa e al servizio, del numero medio di ordini di ogni cliente e di quanti clienti risultino insoddisfatti.

Alla pressione del bottone "simula" l'applicazione svolgerà tante simulazioni dell'evento quante sono quelle da noi specificate e per ognuna salverà le informazioni sopra descritte; nell'interfaccia verrà poi presentata, per ogni variabile, la sua media, così da poter avere dei dati maggiormente fedeli rispetto allo svolgere una sola simulazione.

3. DESCRIZIONE DEI DATA-SET UTILIZZATI

Il data-set utilizzato si compone di tre tabelle: Evento, Piatto e Bevanda. Ogni tabella è caratterizzata da alcuni parametri che andremo a vedere nel dettaglio.

Evento:

- Id (int): codice identificativo univoco;
- Costo (int): costo per organizzare e avviare l'evento (in euro);
- Budget_medio (int): budget che possiede ciascun cliente che prende parte all'evento;
- Partecipanti (int): numero di partecipanti alla manifestazione;
- Descrizione (varchar): descrizione dell'evento.

Bevanda:

- Id (int): codice identificativo univoco;
- Prezzo (double): prezzo di vendita al cliente;
- Costo (double): costo della bevanda per l'azienda;
- Descrizione (varchar): nome della bevanda.

Piatto:

- Id (int): codice identificativo univoco;
- Prezzo (double): prezzo di vendita al cliente;
- Costo (double): costo del piatto per l'azienda;
- Quantità (int): con questo attributo si intende la quantità di prodotto finale che si può vendere producendo un'unità dell'ingrediente principale; è rilevante solo in alcuni piatti, negli altri è settata a 0 (ad esempio se si cucina un bollito misto si stima che si riescano a servire 80 porzioni di "bollito misto con bagnet"; se si supera quel numero di porzioni dovrà essere prodotta un'unità in più di bollito misto);
- Tempo_prep (int): tempo di preparazione del piatto (come già detto prima gli ingredienti si stimano già pronti, questo tempo espresso in secondi si riferisce all'assemblaggio momentaneo degli ingredienti);
- Descrizione (varchar): descrizione del piatto;
- Genere (varchar): categoria di appartenenza del piatto, possibili valori sono antipasto, principale, formaggio e dolce.

Di seguito allego qualche immagine di esempio di alcune righe per ogni tabella, per rendere più comprensibile tutto il discorso.

 id	prezzo	costo	descrizione
0	2,5	1	Vino rosso - Barbera
1	2,5	1	Vino rosso - Dolcetto
2	5	2	Vino rosso - Nebbiolo

Figura 1: Tabella bevande

 id	costo	budget_medio	partecipanti	descrizione
0	10.000	35	1.200	Concerto band rock
1	200	22	100	Evento culturale
2	5.000	40	600	Raduno motociclisti

Figura 2: Tabella eventi

 id	costo	prezzo	quantita	tempo_prep	descrizione	genere
0	2,5	6	000	15	Tagliere di salumi	antipasto
1	1,5	4	000	15	Carne cruda	antipasto
2	0,5	2	000	5	Empanadas	antipasto
3	0,5	2	000	20	Patatine Fritte	contorno
4	0,3	1	000	15	Focaccia	antipasto
5	2,5	6	150	60	Panino pulled pork	principale

Figura 3: Tabella piatto

4. DESCRIZIONE DELLE STRUTTURE DATI E DEGLI ALGORITMI UTILIZZATI

L'applicazione è stata sviluppata in linguaggio Java con il supporto degli applicativi JavaFX per l'interfaccia, attraverso il software SceneBuilder; sono inoltre implementati il pattern DAO (Data Access Object) e il pattern MVC (Model View Controller).

Il programma contiene tre diversi package:

- Db:

contiene le classi ConnectDB e ProvaFinaleDAO, si occupa delle interazioni con il database;

- Tesi:

contiene Il foglio FXML relativo all'interfaccia, il controller di essa, un foglio di stile CSS e la classe Main;

- Model:

contiene le classi che si occupano della logica applicativa del software. La classe Model comprende il codice utilizzato per la simulazione; accanto ad essa troviamo le classi relative agli oggetti utili nello sviluppo dell'applicazione (Piatto, Bevanda, Cliente, Dipendente, Manifestazione), la classe Evento necessaria per la simulazione e la classe Simulazione che, riferendosi al Model, lancia più simulazioni della stessa manifestazione ed effettua una media pesata di tutti i parametri d'interesse, aiutando ad avere dati maggiormente precisi.

La classe Model si occupa di gestire la simulazione vera e propria; innanzitutto crea i clienti e i dipendenti basandosi rispettivamente sul numero medio dei clienti che prendono parte alla manifestazione e sul tempo medio di riordino degli stessi e sul numero di dipendenti inseriti dall'utente nell'interfaccia.

```
public void creaDipendenti(int cassa, int bancone, double paga) {  
  
    listaDipendentiBancone.clear();  
    listaDipendentiCassa.clear();  
  
    for(int i = 0; i < cassa; i++) {  
        Dipendente d = new Dipendente(i, paga);    //un dipendente fisso bevande, tra quelli al bancone  
        listaDipendentiCassa.add(d);  
    }  
    for(int i = cassa; i < (bancone+cassa); i++) {  
        Dipendente d = new Dipendente(i, paga);  
        listaDipendentiBancone.add(d);  
    }  
}
```

```

public void creaClienti(int tempo) {

    listaClienti.clear();
    listaClientiCoda.clear();
    listaClientiDaServire.clear();

    int numero = m.getNumeroPersone();
    int probabilita = r.nextInt(2);
    if(probabilita == 0)
        numero += numero*((r.nextInt(40)+1)/100);
    if(probabilita == 1)
        numero -= numero*((r.nextInt(40)+1)/100);

    System.out.println("numero clienti: "+numero);

    for(int i = 0; i < numero; i++) {
        int prob2 = r.nextInt(2);

        double tmp = 0;
        if(prob2 == 0)
            tmp = tempo+tempo*(r.nextInt((int)tempo/3))/100;
        if(prob2 == 1)
            tmp = tempo-tempo*(r.nextInt((int)tempo/3))/100;

        Cliente c = new Cliente(i,m.getBudgetMedio(),(int)tmp);

        listaClienti.add(c);
        idMapClienti.put(i, c);
    }
}

```

Dopo aver fatto ciò passa all’inizializzazione che, dopo aver ricevuto una lista di piatti e di bevande le memorizza, divide i piatti principali da quelli secondari in altre due liste e infine si occupa di dividere i clienti in tre fasce di arrivo in base all’orario (19/20, 20/21, 21/22); di seguito viene mostrata la prima parte del codice della funzione.

```

public void init(List<Piatto> lp, List<Bevanda> lb) {

    this.listaBevande = lb;
    this.listaPiatti = lp;

    for(Piatto p : lp) {
        if(p.getGenere().equals("principale")) {
            listaPiattiPrincipali.add(p);
        } else {
            listaPiattiSec.add(p);
        }
    }
    queue.clear();
    bevandeOrdinate.clear();
    piattiOrdinati.clear();

    int numero = listaClienti.size(); //divido i clienti
    int primaFascia = numero/3; //19-20
    int secondaFascia = numero/2; //20-21
    int terzaFascia = numero/6; //21-22

    System.out.println("fasce: "+primaFascia+" "+secondaFascia+" "+terzaFascia);

    for(int i = 0; i < primaFascia; i++) {
        double budget = listaClienti.get(i).getBudget();
        int probabilita = r.nextInt(2);
        if(probabilita == 0)
            budget += budget*(r.nextInt(30)+1)/100;
        if(probabilita == 1)
            budget -= budget*(r.nextInt(30)+1)/100;

        listaClienti.get(i).setBudget(budget);

        int intervalloTempo = 60*60/primaFascia; //in secondi

        Evento e = new Evento(oraInizio.plusSeconds(intervalloTempo*i),TipoEvento.ARRIVO_CODA,listaClienti.get(i));
        queue.add(e);
    }
}

```

Dopo l'inizializzazione, si passa poi alla funzione run() che estrae da una PriorityQueue gli eventi (tre sono quelli disponibili: ARRIVO_CODA, ORDINE, SERVIZIO) e li gestisce ognuno in modo diverso con l'obiettivo di portare a completamento la simulazione come descritto al punto 2 della relazione.

Di seguito viene mostrato il primo pezzo del codice della funzione, relativo alla gestione dell'evento ARRIVO_CODA e dell'evento ORDINE nel caso in cui il cliente stia ordinando per la prima volta.

```
public void run() {  
    this.bevandeOrdinate.clear();  
    this.piattiOrdinati.clear();  
  
    while (!queue.isEmpty()) {  
        Evento ev = queue.poll();  
  
        if(ev.getOra().isBefore(oraFine)) {  
            switch(ev.getTipo()) {  
  
                case ARRIVO_CODA:  
  
                    int personeInCoda = listaClientiCoda.size(); //stimo 30 secondi ciascuno +-10  
                    double secondiAttesa = personeInCoda*(30+r.nextInt(11)) / listaDipendentiCassa.size();  
  
                    ev.getCliente().setTempoAttesaCassa(secondiAttesa);  
  
                    if(secondiAttesa > tempoMaxAttesaCassa)  
                        tempoMaxAttesaCassa = (int)secondiAttesa;  
  
                    if(secondiAttesa > maxAttesaCassa) {  
                        if(!ev.getCliente().isInsoddisfattoCoda())  
                            ev.getCliente().setInsoddisfattoCoda();  
                    }  
  
                    listaClientiCoda.add(ev.getCliente());  
  
                    Evento e1 = new Evento(ev.getOra().plusSeconds((int)secondiAttesa), TipoEvento.ORDINE, ev.getCliente());  
  
                    idMapClienti.remove(ev.getCliente().getId());  
                    idMapClienti.put(ev.getCliente().getId(), ev.getCliente());  
  
                    if(e1.getOra().isBefore(oraFine))  
                        queue.add(e1);  
  
                    break;
```

case **ORDINE**:

```
int bevande = listaBevande.size();
int piatti = listaPiatti.size();

int piattiPrinc = listaPiattiPrincipali.size();

listaClientiCoda.remove(ev.getCliente());

if(ev.getCliente().isPrimoOrdine()) { //piatto principale e bevanda

    Piatto p = new Piatto();
    p = listaPiattiPrincipali.get(r.nextInt(piattiPrinc));
    Bevanda b = new Bevanda();
    b = listaBevande.get(r.nextInt(bevande));

    double spesa = p.getPrezzo()+b.getPrezzo();

    if(ev.getCliente().getBudget() > (spesa)) {
        ev.getCliente().decrementaBudget(spesa);
        ev.getCliente().addListaBevande(b);
        ev.getCliente().addListaPiatti(p);

        piattiOrdinati.add(p);
        bevandeOrdinate.add(b);

        incasso += spesa;

        double attesa = 0;

        for(int i = 1; i < listaClientiDaServire.size(); i++) {
            attesa += listaClientiDaServire.get(i).getListaPiatti().get(0).getTempoPrep();
        }
        attesa = attesa/(listaDipendentiBancone.size()-1); //uno per le bevande

        attesa += p.getTempoPrep();

        if(attesa > maxAttesaServizio) {
            if(!ev.getCliente().isInsoddisfattoServizio())
                ev.getCliente().setInsoddisfattoServizio();
        }
        if(attesa > tempoMaxAttesaServizio)
            tempoMaxAttesaServizio = attesa;

        ev.getCliente().setTempoAttesaServizio(attesa);
        ev.getCliente().addNumOrdini();

        listaClientiDaServire.add(ev.getCliente());

        Evento e2 = new Evento(ev.getOra().plusSeconds((int)attesa), TipoEvento.SERVIZIO, ev.getCliente());

        idMapClienti.remove(ev.getCliente().getId());
        idMapClienti.put(ev.getCliente().getId(), ev.getCliente());

        if(e2.getOra().isBefore(oraFine))
            queue.add(e2);
    }

    break;
```

Nel caso di ordine già effettuato, il cliente sceglierà con uguale probabilità tra l'ordinare un piatto, una bevanda o entrambi (tenendo conto del fatto che non si possano ordinare più di due piatti principali durante la stessa manifestazione). L'ultimo evento prevede solo il reinserire il cliente in coda basandosi sul tempo di riordino.

È stata poi creata la classe Simulazione che, servendosi della classe Model, interagisce con il controller per prelevare i dati necessari ed effettuare più

simulazioni della manifestazione desiderata per poi restituire al controller i valori medi ricavati da stampare. In seguito viene proposto un estratto del codice della funzione `simula()`.

```
public void simula (Manifestazione man, List<Piatto> p, List<Bevanda> b, int tempoRiordino, int dipCassa,
                  int dipBancone, double pagaOraria) {

    for(int i = 0; i< numeroSimulazioni; i ++) {

        Model m = new Model(man);
        m.creaClienti(tempoRiordino);
        m.creaDipendenti(dipCassa, dipBancone, pagaOraria);

        m.init(p,b);
        m.run();

        for(Cliente c : m.getIdMapClienti().values()) {
            if(c.isInsoddisfattoCoda() && !c.isInsoddisfattoServizio())
                clientiInsoddisfattiCoda++;
            if(c.isInsoddisfattoServizio() && !c.isInsoddisfattoCoda())
                clientiInsoddisfattiServizio++;
            if(c.isInsoddisfattoCoda() && c.isInsoddisfattoServizio())
                clientiInsEntrambi++;
        }

        guadagnoMedio += m.getGuadagno();
        costoMedio += m.getCostoTotale();
        incassoMedio += m.getIncasso();
        numMedioOrdini += m.getNumOrdiniTotale();

        piattiOrdinati.addAll(m.getPiattoOrdinati());
        bevandeOrdinate.addAll(m.getBevandeOrdinate());

        attesaMediaCassa += m.calcolaAttesaCassa(m.getIdMapClienti());
        attesaMediaServizio += m.calcolaAttesaServizio(m.getIdMapClienti());

        for(Piatto pi : piattiOrdinati) {
            if(mappaPiatti.containsKey(pi.getId())) {
                mappaPiatti.get(pi.getId()).setQuantita2(pi.getQuantita());
                mappaPiatti.get(pi.getId()).setQtaDaProdurre2(pi.getQtaDaProdurre());
            }else {
                mappaPiatti.put(pi.getId(), pi);
            }
        }
    }
}
```

```

    for(Bevanda be : bevandeOrdinate) {
        if(mappaBevande.containsKey(be.getId())) {
            mappaBevande.get(be.getId()).setQuantita2(be.getQuantita());
        }else {
            mappaBevande.put(be.getId(), be);
        }
    }
    if(m.getTempoMaxAttesaCassa() > tempoMaxAttesaCassa)
        tempoMaxAttesaCassa = m.getTempoMaxAttesaCassa();
    if(m.getTempoMaxAttesaServizio() > tempoMaxAttesaServizio)
        tempoMaxAttesaServizio = m.getTempoMaxAttesaServizio();
}

guadagnoMedio = guadagnoMedio/numeroSimulazioni;
costoMedio = costoMedio/numeroSimulazioni;
incassoMedio = incassoMedio/numeroSimulazioni;
numMedioOrdini = numMedioOrdini/numeroSimulazioni;
clientiInsoddisfattiCoda = clientiInsoddisfattiCoda/numeroSimulazioni;
clientiInsoddisfattiServizio = clientiInsoddisfattiServizio/numeroSimulazioni;
clientiInsEntrambi = clientiInsEntrambi/numeroSimulazioni;
attesaMediaCassa = attesaMediaCassa/numeroSimulazioni;
attesaMediaServizio = attesaMediaServizio/numeroSimulazioni;

List<Bevanda> lb = new ArrayList<Bevanda>();
List<Piatto> lp = new ArrayList<Piatto>();

for(Piatto pia : mappaPiatti.values()) {
    double qta = pia.getQuantita()/numeroSimulazioni;
    double d = 0;
    if(pia.getQtaPreparazione() != 0) {
        d = qta/pia.getQtaPreparazione();
        double a = Math.floor(d);
        if(d - a > 0) {
            d++;
            d = Math.floor(d);
        }
    }
    pia.setQtaOrdinMedia(qta);
    pia.setQtaProdMedia(d);
    lp.add(pia);
}

```

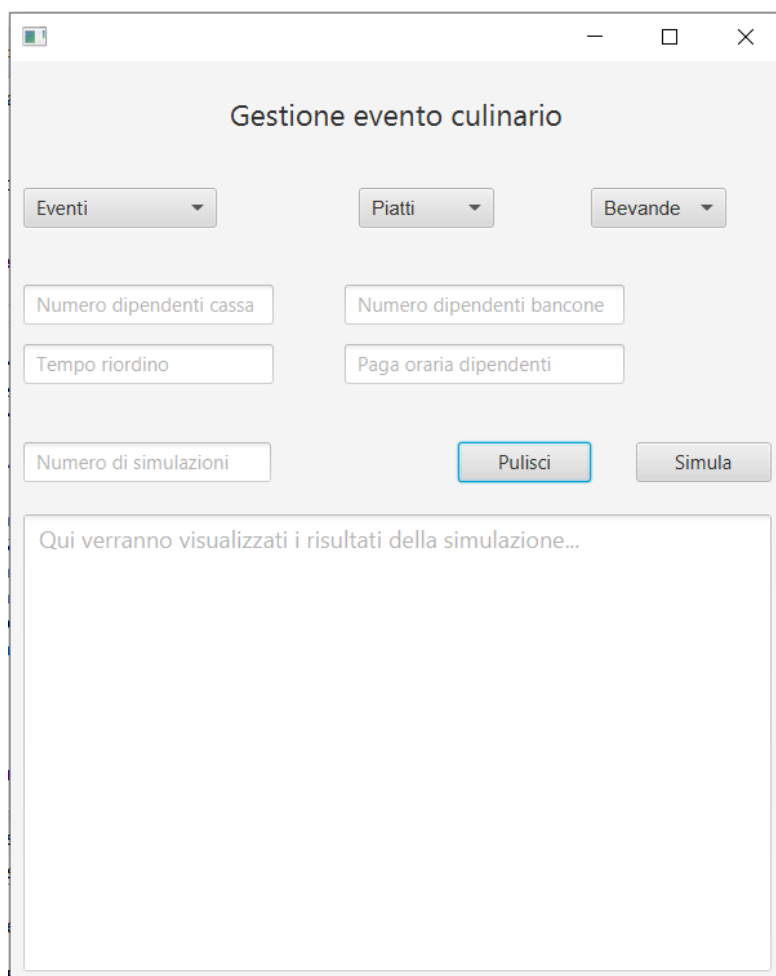
Si tenga conto del fatto che per avere valori significativi nella simulazione si deve inserire un tempo di riordino dei clienti consono a valori reali (almeno 30 minuti), perché se viene inserito un tempo troppo basso si otterrà un forte accumulo di persone sia nella coda per la cassa che nella coda per il servizio, cosa che renderà la simulazione poco rilevante ai fini reali; inoltre, si deve tenere conto del fatto che ogni piatto necessita di una preparazione degli ingredienti che viene fatta prima dell'avvio della manifestazione, quindi per simulazioni a cui non prendono parte grandi numeri di persone ha poco senso avviare tenendo selezionati tutti i piatti.

5. DIAGRAMMA DELLE CLASSI

In allegato al progetto nella cartella documenti si trova un diagramma UML di tutte le classi, creato con UML Lab su Eclipse.

6. VIDEATE DELL'APPLICAZIONE E RISULTATI SPERIMENTALI

All'apertura dell'applicazione viene visualizzata la schermata sottostante; bisogna tenere presente che inizialmente sono selezionati tutti i piatti e le bevande disponibili, l'utente dovrà selezionare una manifestazione (tra parentesi è indicato il numero di persone medio) e inserire gli altri parametri necessari già discussi in precedenza.



The screenshot shows a software window titled "Gestione evento culinario". At the top, there are three dropdown menus labeled "Eventi", "Piatti", and "Bevande". Below these are five input fields: "Numero dipendenti cassa", "Numero dipendenti bancone", "Tempo riordino", "Paga oraria dipendenti", and "Numero di simulazioni". There are two buttons, "Pulisci" and "Simula", positioned to the right of the "Numero di simulazioni" field. At the bottom of the window is a large text area containing the placeholder text "Qui verranno visualizzati i risultati della simulazione...".

Una volta inseriti i parametri, cliccando su simula, verranno stampati a video i risultati della simulazione.

Nell'esempio che segue si è scelto un evento enogastronomico con numero di persone medie 400 e budget medio 30€.

Gestione evento culinario

Evento enog... Piatti Bevande

2 5

45 8

15 Pulisci Simula

Piatti ordinati:

- Panino pulled pork: prezzo 6.0; quantità totale 993; quantità media: 66.0; quantità da produrre media: 1.0
- Panino porchetta: prezzo 5.0; quantità totale 1012; quantità media: 67.0; quantità da produrre media: 1.0
- Panino Hamburger: prezzo 4.5; quantità totale 987; quantità media: 65.0
- Panino salsiccia: prezzo 4.5; quantità totale 968; quantità media: 64.0

Per prima cosa vengono stampati tutti i piatti e le bevande scelte per la simulazione; ogni prodotto è seguito dal prezzo, dalla quantità totale ordinata nelle varie simulazioni e dalla quantità media ordinata in ognuna di esse.

Gestione evento culinario

Evento enog... Piatti Bevande

2 5

45 8

15 Pulisci Simula

Evento : Evento enogastronomico; costo: 1000
clienti insoddisfatti cassa : 3,00
clienti insoddisfatti servizio : 343,00
clienti insoddisfatti cassa e servizio : 13,47
costo medio : 5188,23
guadagno medio : 3646,37
incasso medio : 8834,60
attesa media cassa : 531,81
attesa media servizio : 3504,54
tempo max attesa cassa : 2203
tempo max attesa servizio : 7105,00
numero medio di ordini : 2,10

Dopo i piatti e le bevande vengono stampati i valori di maggiore interesse; da essi l'utente può vedere se ci sia qualcosa da variare al fine di rendere migliore l'evento in termini economici e di soddisfazione della clientela.

Ad esempio, nella figura a fianco si nota che gran parte delle persone sono insoddisfatte del tempo di attesa del servizio; l'utente in questo caso può aumentare il numero di dipendenti che lavoreranno per ovviare a questo problema.

Gestione evento culinario

Evento enog... Piatti Bevande

3 9

45 8

15 Pulisci Simula

Evento : Evento enogastronomico; costo: 1000
 clienti insoddisfatti cassa : 0,00
 clienti insoddisfatti servizio : 11,60
 clienti insoddisfatti cassa e servizio : 0,00
 costo medio : 5665,60
 guadagno medio : 3734,60
 incasso medio : 9400,20
 attesa media cassa : 21,06
 attesa media servizio : 874,28
 tempo max attesa cassa : 400
 tempo max attesa servizio : 1993,75
 numero medio di ordini : 2,27

al contrario, in questo caso si può notare che aumentando i dipendenti si riesce a ridurre l'insoddisfazione e si generano più ricavi e più guadagni (a fronte di un aumento dei costi del personale abbiamo un aumento delle vendite), che sono sicuramente dei fattori di grande rilievo per l'azienda.

L'ultimo pulsante presente che non è ancora stato menzionato è il pulsante "Pulisci" che permette all'utente di far tornare l'interfaccia grafica nella condizione iniziale, ovvero con tutti i piatti e le bevande selezionate e con i campi vuoti.

Link al video dimostrativo su YouTube:

7. VALUTAZIONE DEI RISULTATI E CONCLUSIONI

L'obiettivo dell'applicazione è quello di essere effettivamente usato per capire quale evento all'aperto organizzare in base alle esigenze e ai fattori limitanti (tempo che si può spendere per organizzarlo, dipendenti disponibili, costo di avvio).

Le tempistiche di risoluzione della simulazione dipendono fortemente dal numero di persone che prendono parte all'evento e dal numero di simulazioni consecutive da effettuare: se per numeri di persone inferiori a 500 e con meno

di 20 simulazioni consecutive la risoluzione è pressoché istantanea, scegliendo ad esempio di svolgere 10 simulazioni del concerto rock che ha 1200 partecipanti medi bisognerà attendere qualche secondo prima di vedere i risultati stampati a video.

Dal mio punto di vista l'applicazione è rilevante per il titolare dell'agriturismo anche se per renderla più fedele alla realtà bisognerà organizzare diversi eventi e compiere qualche studio su di essi per valutare le preferenze delle persone al fine di rendere le scelte dei piatti maggiormente veritiere (al momento la scelta del piatto è casuale, quindi in media verranno ordinate quantità simili di tutti i piatti).

L'obiettivo è comunque quello di portare avanti il progetto e ampliare l'applicazione per aggiungere altre informazioni utili, per esempio pianificando la produzione degli ingredienti in base alle quantità finali consumate o la possibilità di pianificare eventi all'interno con piatti che richiedono di essere cucinati da cuochi professionisti e non solo assemblati sul momento.

Infine, bisogna sempre tenere conto che le variabili in gioco quando si organizza una manifestazione di questo tipo sono molteplici e questo programma si propone solo come uno strumento di aiuto e di valutazione, e che la sua reale veridicità nel simulare un evento sarà da testarsi successivamente comparando i dati reali ottenuti con quelli sperimentali.



Quest'opera è stata rilasciata con licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale.

Copia della licenza consultabile al sito web: <http://creativecommons.org/licenses/by-nc-sa/4.0/>