TEHNICI FUNDAMENTALE DE PROGRAMARE

Aplicație de comandat mancare

*-JavaSE15, Java Swing-*

Realizat de:

Tanul Gabriel-Ștefan

an 2 grupa 30222

1. Problemă și soluție
2. Obiective
3. Analiză
   1. Pachete
      1. Model
      2. Business layer
      3. Data layer
      4. Presentation Layer
4. Dezvoltare grafică
5. Interogări MySQL
6. Serializare și lambda
7. Posibilități de dezvoltare ulterioară

1. Problemă și soluție

Se considera o aplicație de comenzi de mancare. Se cere utilizarea bazelor de date relationale pentru stocarea datelor legate de **produse,** **utilizatori și comenzi realizate de clienti.**

Este costisitor din punct de vedere al timpului să ținem evidența multor date pe foaie, cum se obișnuia acum mulți ani când nu aveam la dispoziție dispozitive de calcul și sisteme hardware avansate ca în prezent, iar o soluție bună la această problemă ar fi implementarea unei aplicații cu interfață grafică într-un limbaj de programare cunoscut care să folosească o bază de date unde se pot reține informațiile despre un anumit subiect sau în cazul de față despre comenzile unui depozit mult mai ușor. Mai mult de atat, in cazul acestei aplicatii este mai simplu de a plasa o comanda deoarece se poate lega totul la un website online si persoanele care se inregistreaza pot mai apoi sa vizualizeze un meniu si sa comande mancare. În cazul de față am folosit limbajul de programare Java și interfață grafică din Java Swing folosind mediul de dezvoltare **IntelliJ IDEA, iar baza de date utilizată a fost implementată în Navicat pe un server de MySQL.**

2. Obiective

Scopul principal al acestui proiect a fost de a dezvolta și implementa o aplicație unde cei desemnați să se ocupe cu gestiunea comenzilor pot elibera mai simplu o chitanta a unei comenzi deoarece cei care le plaseaza sunt clientii. In aplicație sunt mai multe tipuri de utilizatori, **admin, agent și client. Adminul** are acces la toate datele legate de toti utilizatorii și este cel care se ocupa de introducerea produselor în baza de date care ulterior pot fi vizualizate în timp real de către persoane care doresc sa plaseze o comanda. **Agentul** este responsabil de transmiterea comenzii către bucătărie pentru sa se servi și mai apoi către curierul care va livra mâncarea, estimand ora de ajungere la destinație. **Clientul** vizualizează meniul disponibil, alege produsele, și apoi confirma comanda pentru a fi transmisă către centrală și servită.

**Cum să implementez soluția?**



**Utilizare**: modificare date, adăugare date, ștergere date

**Principalul actor: utilizatorul, clientul, adminul**

**Scenariul cel mai probabil:**

1. În meniul principal, actorul selecteaza Logare sau Inregistrare
2. În cazul în care se selectează înregistrare, este doar pentru clienți, iar după ce se introduc datele corect, se afisează un mesaj de confirmare în cazul în care înregistrarea are loc cu succes.
3. Revenire la meniul principal și se selectează logare, după care se introduce e-mail și parolă.
4. Dacă logarea are loc cu succes și este executată de un client, se deschide pagina de plasare a unei comenzi.
5. Dacă logarea este executată de un admin atunci se deschide pagina de editare/ștergere/modificare a utilizatorilor și a produselor existente.
6. Dacă logarea este executată de un agent atunci se deschide pagina de eliberare chitanță/ștergere comandă în cazul în care clientul comunică anularea acesteia.
7. Calculatorul verifică corectitudinea datelor și le adaugă în baza de date.

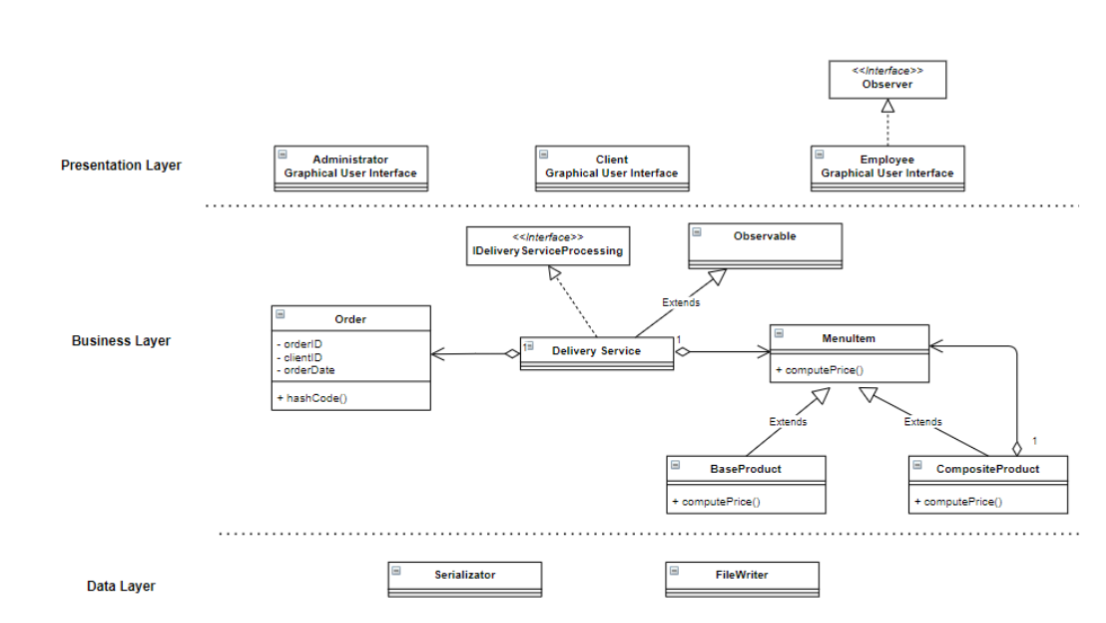
**Scenariul mai puțin probabil:** Date introduse incorect

* Utilizatorul introduce datele greșit (credențiale greșite)
* În acest caz, se revine la pasul 3.

1. Analiză

Pentru a reliza o conexiune în timp real cu o bază de date este nevoie de un conector JDBC care să permită transmiterea datelor înspre server folosindu-se de user, parola și url-ul serverului.

De asemenea am ales o structurare a pachetelor din java după cum urmează.



a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a

a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a

a a a a a a a a a a a a a a a a a a a a a a a a a

În **Model** am ales ca variabile instanță clasele implementate pentru frame, url, e-mail și parolă de la server și alte frame-uri legate de erori. Aici am adăugat clasele **ExtragereAntet, ExtragereDate, ExecProc, DBMethods, ClockPane, Model, Table.**

**ExtragereDate** este o clasa ce conține metode care extrag datele dintr-un anumit tabel transmis ca parametru funcției, funcție care folosește biblioteca java.SQL și care prin intermediul conectorului JDBC comunică cu baza de date MySQL și preia informațiile dorite. De asemenea

**ExtragereAntet** este bazată pe același principiu însă aici este returnat antetul unui tabel pentru folosirea acestuia în interfața grafică. **ExecProc** execută o anumită interogare MySQL și în funcție de **result set-ul** acesteia se transmite la un ***String***care se poate folosi mai departe în aplicație ( indiferent că este vorba de antetul unui **result set** sau datele acestuia).

**ClockPane** este un panel care afișează în timp real data în colțul frame-urilor.

**Table** este o clasa implementată de mine care conține mai multe funcționalități ale JTable-ului.

**DBMethods** este o interfață care conține metode generice legate de accesul datelor din baza de date (**ștergere, modificare, adăugare)** care este implementată de majoritatea claselor care modifică informații în baza de date.

**În Business Layer** am adăugat clasele **BusinessLogic, Main, DeliveryService, IdeliveryServiceProcess, Menu, Order.** Sunt toate clasele responsabile de controlul datelor și informațiilor care intră și ies din sistemul aplicației respectiv din interfața grafică.

Pachetul **Data access** am adăugat clasele **DBConnection, DBUpdate, ClientData, AdminData, OrdersData** care realizează serializarea datelor. În **DBConnection** se realizează prin **DriverManager** conexiunea cu serverul MySQL având URL-ul și parola serverului folosit iar separat este declarat numele bazei de date utilizate. **DataUpdate** are toate action-listener-ele butoanelor de introducere, editare și stergere a datelor din tabelele existente iar după cum spune și numele, aici se realizează actualizarea bazei de date, a tabelelor din interfața grafică și ale **InputStream-urilor** și **OutputStream-urilor**. Se sterg, introduc sau actualizează datele prin intermediul **PreparedStatement-urilor sau CallableStatement-urilor** care accesează proceduri existente din baza de date sau execută un query dat ca parametru funcțiilor. De asemenea pe partea de serializare se folosesc **BufferedStreamReader, FileWriter, etc.**

**ClientData, AdminData, OrdersData** sunt clasele care conțin metode implementate din interfața **DBMethods** și care reprezintă prelucrarea datelor pentru baza de date și pentru fișierele de output/input.

**PresentationLayer** este pachetul care conține toate clasele interfeței grafice **ClientsFrame, Login, Frame, RegisterFrame, MeniuPrincipal, OrdersFrame, AdminFrame.** Fiecare dintre acestea extinde clasa JFrame și creează câte o fereastră pentru fiecare cu o funcționalitate diferită și interfață diferită.

În **ClientsFrame** se află tabelul cu datele despre produse, selectoare pentru plasarea comenzii și text-box-uri pentru afișarea totalului comenzii, butoane de confirmare etc.

**OrdersFrame** conține tabelul cu comenzile noi plasate, date despre agentul care s-a logat pentru procesarea comenzii și butoane de emitere chitanță și ștergere comandă.

**LoginFrame** reprezintă pagina principală de logare unde după e-mail și parolă utilizatorii accesează paginile aplicației.

**RegisterFrame** este dedicat clienților care doresc să își creeze un cont pe platformă și intenționează să plaseze comenzi.

1. Dezvoltarea grafică

* SWING API
  1. Parte din clasele fundamentale java (JFC)
  2. Facilitează scrierea grafică a aplicațiilor java
  3. Include 17 pachete standard printre care cel mai folosit:
* **javax.swing** - Is the most important package from Swing

|  |  |
| --- | --- |
| **Tipul componentei** | **Examples** |
| Componente atomice | JLabel, JButton, JCheckBox, JRadioButton, JToggleButton, JScrollBar, JSlider |
| Componente complexe | JTable, JTree, JComboBox, JList, JFileChooser, JColorChooser, JOptionPane |
| Componente de tip text | JTextField, JPasswordField, JTextArea, JEditorPane, JTextPane |
| Meniuri | JMenuBar, JMenu, JPopupMenu, JMenuItem, CheckboxMenuItem , JRadioButtonMenuItem |
| Cointainere intermediare | JPanel, JTabbedPane, JDesktopPane |
| Containere principale | JFrame, JDialog |

Am folosit un container principal de tip JFrame pentru a ilustra fereastra principală a calculatorului iar apoi am creat un fel de ferestre de tip pop-up care apar în cazul excepțiilor și anume a erorilor prezentate mai sus.

Managerul de aspect este folosit pentru aranjarea într-o anumită structură a containerelor sau a componentelor.

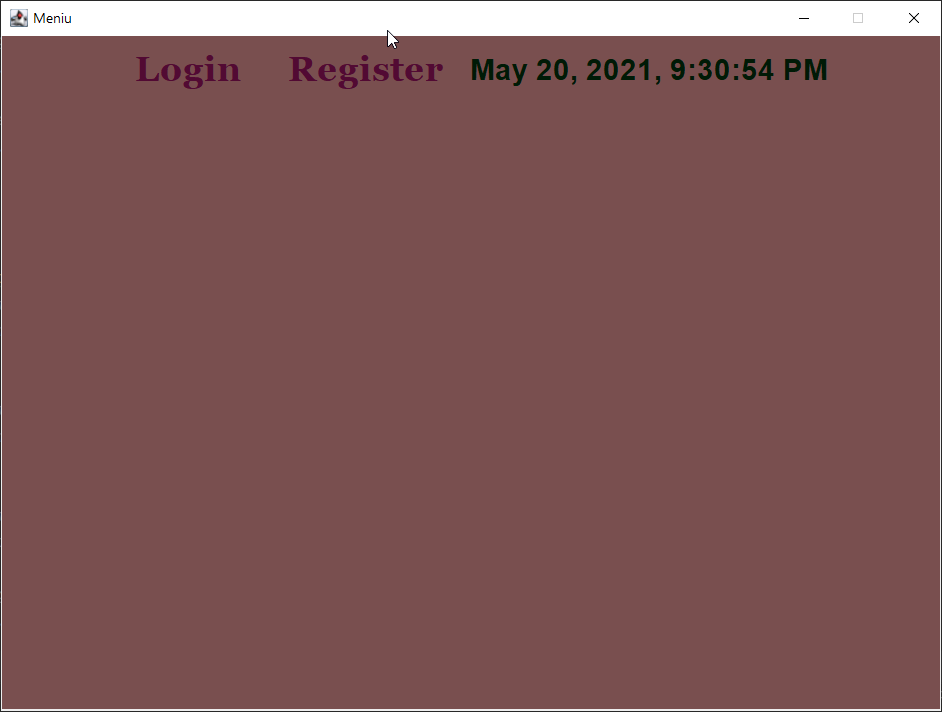
a) BorderLayout – plasează componentele în cinci zone ale containerului principal: sus, jos stânga, dreapta și centru.

b) BoxLayout – plasează componentele pe o linie sau pe o coloană

c) FlowLayout – plasează componentele într-o singură linie dreaptă

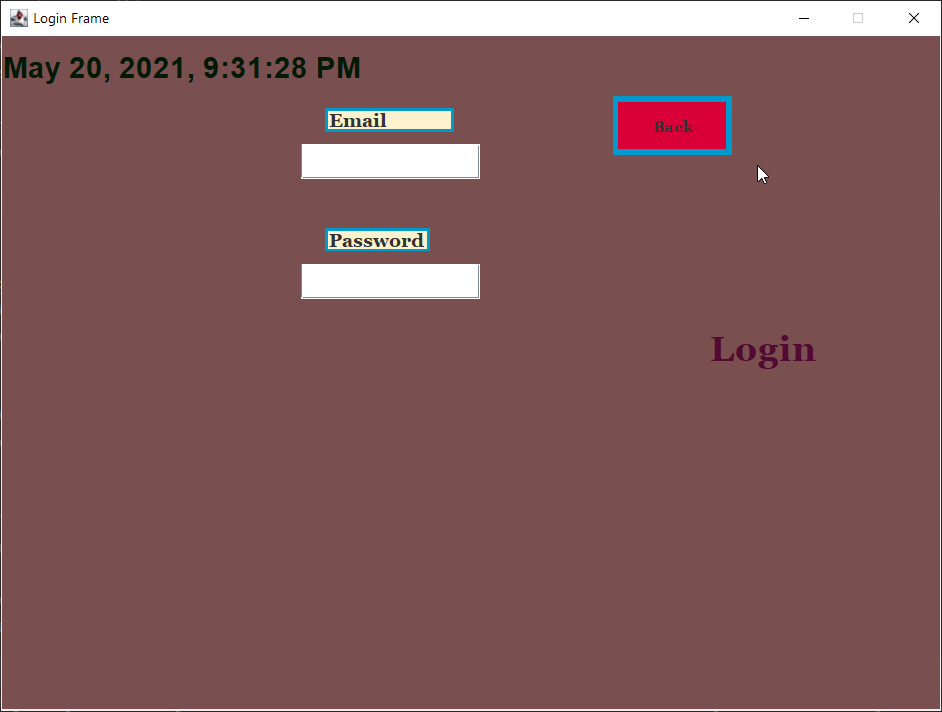
d) GridLayout – plasează componentele într-un aspect matricial în mai multe celule

Am ales un GridLayout însă nu l-am folosit pe cel implicit ci am preferat să aleg eu coordonatele pentru componente deoarece a fost mai simplu să le așez exact în locul dorit.

Așa arată partea de grafică a calculatorului. Au fost implementate butoane pentru a alege operația dorită, pentru a confirma introducerea polinoamelor și etichete pentru a ilustra rezultatul sau restul polinomului final.

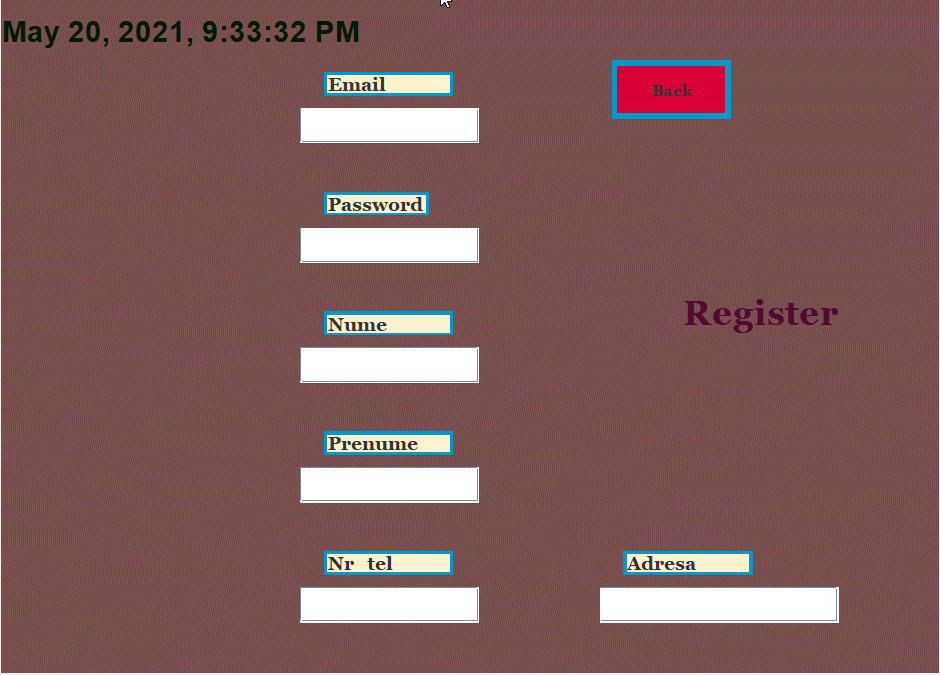
a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a

Butoane ferestre



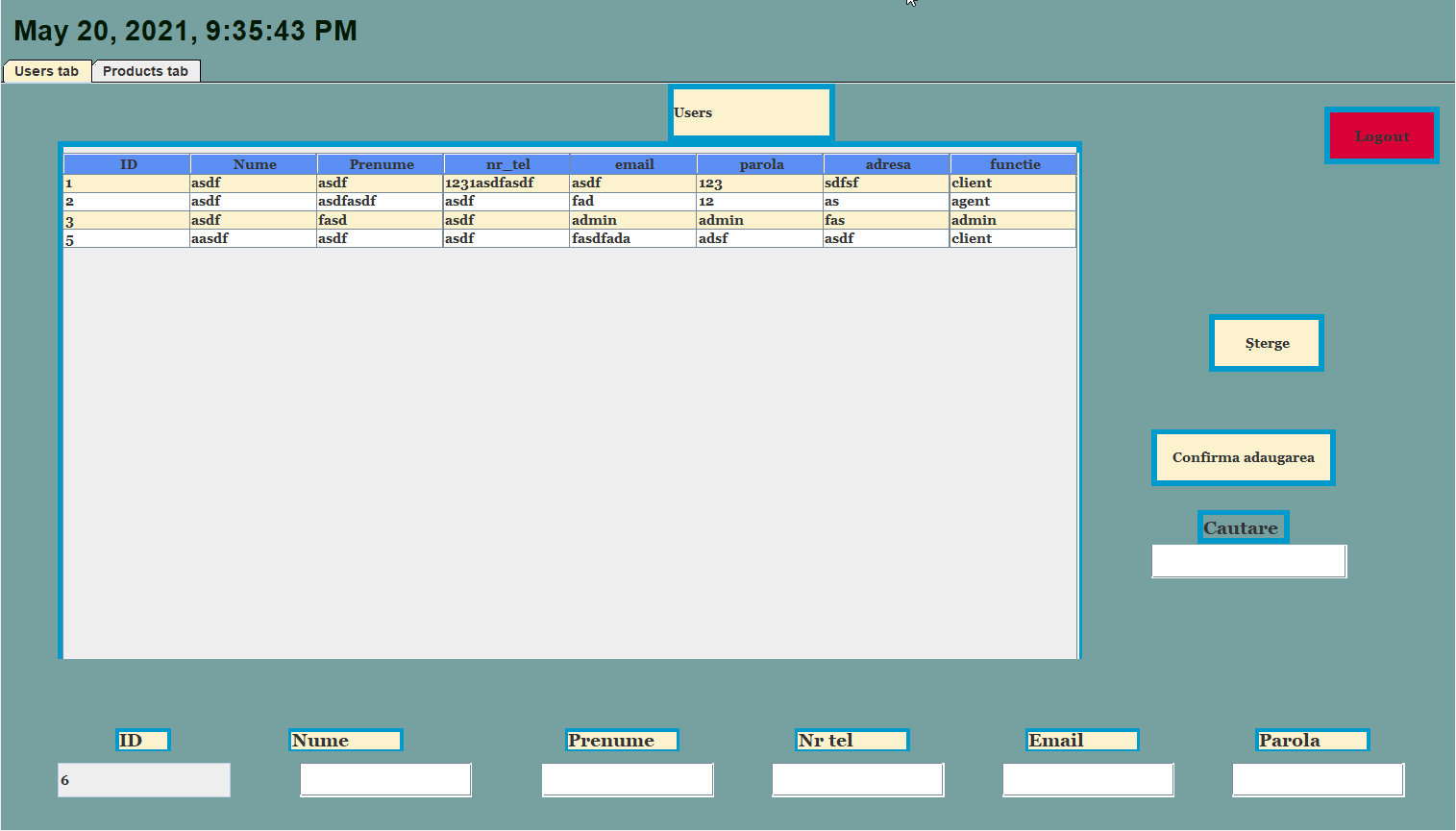
Butoane de decizie

Credențiale pentru acces în aplicație



Butoane de decizie

Credențiale pentru creare de cont



Date tabel

Butoane de decizie

Text-field-uri date intrare

1. Interogări în MySQL

Pentru a modifica datele din server m-am folosit și de anumite proceduri stocate pe server care prin intermediul **CallableStatement-ului** din java se accesau procedurile și se manipulau datele din server ( stergere, actualizare și introducere). De exemplu, proceduri precum **insertOrder, deleteOrder** preiau datele transmise ca parametru și se actualizează în tabel.

1. Serializare și lambda

Serializare

◦ Mecanism de conversie a stării unui obiect într-un flux de octeți

Deserializarea

◦ Fluxul de octeți este utilizat pentru a recrea obiectul Java real în memorie

• Pentru a face un obiect Java serializabil, implementăm interfața java.io.Serializable

• Clasa ObjectOutputStream conține metoda writeObject () pentru serializarea unui obiect

◦ public final void writeObject (Object obj) aruncă IOException

• Clasa ObjectInputStream conține metoda readObject () pentru deserializarea unui obiect

◦ public final Object readObject () aruncă IOException, ClassNotFoundException

De asemenea pentru filtrari am folosti expresii precum stream.filter().collect(), si Collections.toList() pentru a filtra liste de anumite obiecte.

Scrierea în fișiere a fost realizată cu FileWriter și serializarea și deserializarea cu ObjectOutputStream și ObjectInputStream.

1. Posibilități de dezvoltare ulterioară

Ulterior se pot alege anumite elemente grafice care sa usureze utilitatea acestei aplicatii. Se pot face triggere și anumite listene-re pentru a ușura alegerea datelor din text-field-uri dar și pentru a modifica automat datele din baza de date. Se pot adăuga imagini cu fiecare tip de mâncare existent pentru a oferi o sugestie de prezentare produselor și a face mai ușoară deciderea clienților. De asemenea se pot adauga tabele care să stocheze datele despre chitante, facturi, cheltuieli și profit în cadrul restaurantului.

Bibliografie

[1] https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

[2] https://docs.oracle.com/javase/8/docs/api/java/util/Timer.html

[3] http://www.tutorialspoint.com/java/util/timer\_schedule\_period.html

[4] http://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-andthreadpoolexecutor.html

[5] B. Goetz et al., Java Concurrency in Practice, Addison-Wesley Professional; 1 edition (May 19, 2006)

[6] [Serializable (Java Platform SE 8 ) (oracle.com)](https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html)

[7] [Hash table - Wikipedia](https://en.wikipedia.org/wiki/Hash_table)