

Üzleti igény indokoltságának ismertetése:

Jelenlegi helyzet összefoglalása:

- A probléma a programozás 3 tantárgyból indult, aminek keretein belül egy szoftver tervezési és megvalósítási folyamatot kell végrehajtani. Napjainkban a pandémia idejében egyre gyakoribbá válik az online termékek vásárlása, és ennek a céljából készítettünk egy alkalmazást, aminek a segítségével a mindennapi vásárlás megkönnyebbíthetjük akár otthonról is szoftverünkkel, ugyan is ez egy különböző termékvásárlás lebonyolítását segítő szoftver, ami segíti az eladást cégeknek, boltoknak, szupermarketeknek és milliómás fogyasztás eladni kívánó szolgáltatóknak, amiket később implementálni tudnak saját üzletükben vagy weboldalukon, ezzel segítve a fogyasztók vásárlását.

Változtatási igény összegzése:

- Manapság egyre gyakoribb az online vásárlás ennek okai, hogy sokkal egyszerűbb, gyorsabb és probléma mentesebb, mint a lokális vásárlás. Mindenki előfordul, hogy felírja mit szeretne venni, mielőtt elmegy a boltba, azonban megesik, hogy otthon felejtjük a kis cetlinket vagy kimaradt róla egy fontos termék, azonban ha online a szoftverünket használja pontosan nyomon tudja követni, hogy hol tart a vásárlásnál és lemaradt terméket is gyorsan gond nélkül megtudja venni akár később is, hiszen nincs más dolga csak rányomni a vásárlás gombra, nem beszélve arról, mielőtt boltba indulunk fogalmunk sincs, hogy milyen termékek szerepelnek a boltba, így sajnos van, hogy feleslegesen boltról boltra járva keressük a terméket, amit megszeretnénk vásárolni, viszont a szoftverünkben nyomon tudja követni a termékek mennyiségét, így ez sose fordul elő.

Változtatás üzleti eredménye:

- Az alkalmazás fejlesztésének és tervezésének két fontos szerepe van, egyrésztől saját magunknak egy gyakorlás az órán tanult JAVA programtervezés megvalósítására és gyakorlására, ami tartalmazza a szoftver fejlesztési folyamatokat, a szoftver tervezést és az adatbázis kapcsolatát magával a szoftverrel, illetve a csapat munkát és a különböző folyamatok összehangolását. Más részről pedig egy olyan szoftver prototípusának megvalósítása, amit később akár több szolgáltatók is tudnak implementálni a saját rendszerükbe ezzel segítve a fogyasztókat.

Funkcionális és működési követelmények:

Kért új funkciók és szolgáltatások meghatározása, részletes bemutatása:

- Az alkalmazás JAVA programozási nyelven, NetBeans fejlesztői környezetben és MySQL adatbázis segítségével íródott. A fejlesztés, mint említettem egy prototípus, így valós környezetben való implementálásához még további folyamatok és fejlesztések szükségesek, amiket a megrendelővel kell előre egyeztetni.

JTABLE

What you want buy	How much	What the shop sells	How much the Shop has	Did you manage to buy it?	How much could you buy

Itt fogjuk majd látni, hogy:



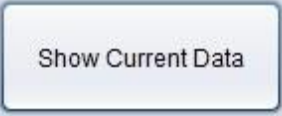
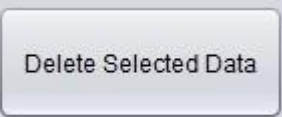
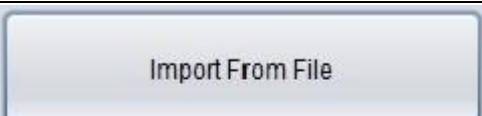
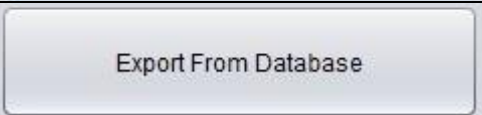
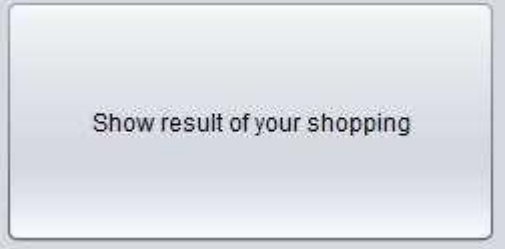
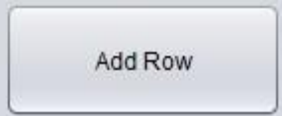
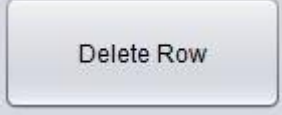
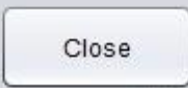
- mit akarunk venni
- mennyit tudunk
- mit árul a bolt
- mennyit terméket árul az adott termékből
- megtudtuk-e vásárolni a terméket
- mennyit tudunk vásárolni az adott termékből

JTEXTFIELD

A JTable-ben megtalálja a keresett szavat.

Betudsz írni egy szöveget a kiválasztott elembe

JBUTTON

	A beírt szöveget megkeresi
	Menti a változtatásokat a táblázatban
	Megmutatja a mostani állapotot
	Törli a kiválasztott Elemet
	Importálunk egy csv.fájlt a fájlok közül
	Exportáljuk az adatokat az adatbázisból
	Megmutatja a vásárlás eredményét
	A táblázatba hozzá ad egy új sort
	Töröl egy sort a táblázatból
	Kilép az alkalmazásból

KÓD

```
1  */
2  package com.mycompany.java_labor_projekt;
3
4  import com.opencsv.*;
5  import java.awt.*;
6  import java.awt.event.WindowEvent;
7  import java.io.*;
8  import java.sql.Connection;
9  import java.sql.DriverManager;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.sql.Statement;
13 import java.util.*;
14 import javax.swing.JOptionPane;
15 import javax.swing.table.DefaultTableModel;
16
17 public class MainFrame extends javax.swing.JFrame {
18
19     /**
20      * Creates new form MainFrame
21      */
22     private DefaultTableModel dt;
23     private String nextcell = "";
24     private final String nothing = "";
25     private String[] s = new String[6];
26     private Statement statement;
27     private ResultSet results;
28     private final String jdbcURL = "jdbc:mysql://localhost:3306/shop_database?useUnicode=true&character_set_server=utf8";
29     private final String username = "root";
30     private final String password = "admin";
31
32     public MainFrame() {
33         initComponents();
34     }
35
36     /**
37      * This method is called from within the constructor to initialize the form.
38      * WARNING: Do NOT modify this code. The content of this method is always
39      * regenerated by the Form Editor.
40      */
41     @SuppressWarnings("unchecked")
42     Generated Code
43
44     private void btn_saveActionPerformed(java.awt.event.ActionEvent evt) {
45         // A kiválasztott cellában állva tud adatot módosítani/hozzáadni a táblába és az adatbázisba
46         String before = "";
47         try {
48             Connection connection = DriverManager.getConnection(jdbcURL, username, password);
49             int index = (int) table.getSelectedColumn();
50             if (table.getValueAt(row, table.getSelectedColumn()) == null) {
51                 before = "";
52             } else {
53                 before = table.getValueAt(row, table.getSelectedColumn()).toString();
54             }
55             statement = connection.createStatement();
56             table.setValueAt(table.getValueAt(row, table.getSelectedColumn()), row, table.getSelectedColumn());
57             if (index == 0) {
58                 nextcell = table.getValueAt(row, table.getSelectedColumn() + 1).toString();
59                 statement.executeUpdate("UPDATE Main_table SET List_Items = '" + txt_input.getText() + "' WHERE List_Items_DB LIKE '" + before + "' AND List_Items_DB LIKE '" + nextcell + "' "
60                     + "ORDER BY shop_database.main_table.Rows ASC LIMIT 1;");
61             } else if (index == 1) {
62                 nextcell = table.getValueAt(row, table.getSelectedColumn() - 1).toString();
63                 statement.executeUpdate("UPDATE Main_table SET List_Items_DB = '" + txt_input.getText() + "' WHERE List_Items_DB LIKE '" + before + "' AND List_Items_DB LIKE '" + nextcell + "' "
64                     + "ORDER BY shop_database.main_table.Rows ASC LIMIT 1;");
65             } else if (index == 2) {
66                 nextcell = table.getValueAt(row, table.getSelectedColumn() + 1).toString();
67                 statement.executeUpdate("UPDATE Main_table SET Shop_Items = '" + txt_input.getText() + "' WHERE Shop_Items LIKE '" + before + "' AND Shop_Items_DB LIKE '" + nextcell + "' "
68                     + "ORDER BY shop_database.main_table.Rows ASC LIMIT 1;");
69             } else if (index == 3) {
70                 nextcell = table.getValueAt(row, table.getSelectedColumn() - 1).toString();
71                 statement.executeUpdate("UPDATE Main_table SET Shop_Items_DB = '" + txt_input.getText() + "' WHERE Shop_Items_DB LIKE '" + before + "' AND Shop_Items_DB LIKE '" + nextcell + "' "
72                     + "ORDER BY shop_database.main_table.Rows ASC LIMIT 1;");
73             } else {
74                 JOptionPane.showMessageDialog(parentComponent, null, message: "You cannot change the value there!",
75                     "InfoBox: " + "Disabled Action", messageType: JOptionPane.INFORMATION_MESSAGE);
76             }
77             statement.close();
78             connection.close();
79         } catch (ArrayIndexOutOfBoundsException boundsException) {
80             JOptionPane.showMessageDialog(parentComponent, null, message: "Please click, 'Show Current Data' button, before trying to change the value!", "InfoBox: "
81                 + "Wrong Usages", messageType: JOptionPane.INFORMATION_MESSAGE);
82         } catch (Exception e) {
83             e.printStackTrace();
84         }
85     }
86 }
```

```

private void btn_deleteActionPerformed(java.awt.event.ActionEvent evt) {
    // Torli az aktualis cellaban lévő adatokat a táblából és az adatbázisból
    String actual = "";
    int index = table.getSelectedColumn();
    try {
        Connection connection = DriverManager.getConnection(url:jdbcURL, user:username, password);
        Statement = connection.createStatement();
        if (table.getValueAt(row:table.getSelectedRow(), column:table.getSelectedColumn()) == null) {
            actual = "";
        } else {
            actual = table.getValueAt(row:table.getSelectedRow(), column:table.getSelectedColumn()).toString();
        }
        if (index == 0) {
            nextcell = table.getValueAt(row:table.getSelectedRow(), table.getSelectedColumn() + 1).toString();
            statement.executeUpdate("UPDATE Main_table SET List_Items = '" + nothing + "' WHERE List_Items Like '" + actual
                + "' AND List_Items_DB LIKE '" + nextcell + "'");
            table.setValueAt(aValue:"", row:table.getSelectedRow(), column:table.getSelectedColumn());
        } else if (index == 1) {
            nextcell = table.getValueAt(row:table.getSelectedRow(), table.getSelectedColumn() - 1).toString();
            statement.executeUpdate("UPDATE Main_table SET List_Items_DB = '" + nothing + "' WHERE List_Items_DB Like '" + actual
                + "' AND List_Items LIKE '" + nextcell + "'");
            table.setValueAt(aValue:"", row:table.getSelectedRow(), column:table.getSelectedColumn());
        } else if (index == 2) {
            nextcell = table.getValueAt(row:table.getSelectedRow(), table.getSelectedColumn() + 1).toString();
            statement.executeUpdate("UPDATE Main_table SET Shop_Items = '" + nothing + "' WHERE Shop_Items Like '" + actual
                + "' AND Shop_Items_DB LIKE '" + nextcell + "'");
            table.setValueAt(aValue:"", row:table.getSelectedRow(), column:table.getSelectedColumn());
        } else if (index == 3) {
            nextcell = table.getValueAt(row:table.getSelectedRow(), table.getSelectedColumn() - 1).toString();
            statement.executeUpdate("UPDATE Main_table SET Shop_Items_DB = '" + nothing + "' WHERE Shop_Items_DB Like '" + actual
                + "' AND Shop_Items LIKE '" + nextcell + "'");
            table.setValueAt(aValue:"", row:table.getSelectedRow(), column:table.getSelectedColumn());
        } else {
            JOptionPane.showMessageDialog(parentComponent:null, message:"You cannot delete the value there!",
                "InfoBox: " + "Disabled Action", messageType:JOptionPane.INFORMATION_MESSAGE);
        }

        statement.close();
        connection.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

private void btn_importActionPerformed(java.awt.event.ActionEvent evt) {
    // Egy két oszlopos és akár végtelen soros csv fájlból beimportálja a táblába és az adatbázisba az adatokat
    Frame f = new Frame();
    FileDialog openf = new FileDialog(f, title:"Choose a file");
    openf.setVisible(b:true);
    String[] nextRecord;
    dt = (DefaultTableModel) table.getModel();
    try {
        Connection connection = DriverManager.getConnection(url:jdbcURL, user:username, password);
        InputStream fr = new FileInputStream(openf.getDirectory() + "/" + openf.getFile());
        CSVReader csvReader = new CSVReader(new InputStreamReader(fr, charsetName:"Windows-1250"));
        Statement = connection.createStatement();
        while ((nextRecord = csvReader.readNext()) != null) {
            for (String cell : nextRecord) {
                s = cell.split(sep:";");
                /*statement.executeUpdate("INSERT INTO Main_table ('List_Items','List_Items_DB') " +
                    "VALUES ('"+s[0]+"','"+s[1]+"')");*/
                statement.executeUpdate("INSERT INTO Main_table ('List_Items','List_Items_DB','Shop_Items','Shop_Items_DB') "
                    + "VALUES ('" + s[0] + "','" + s[1] + "','" + s[2] + "','" + s[3] + "')");
            }
        }
        String sql = ("SELECT CONVERT(List_Items USING utf8),"
            + "CONVERT(List_Items_DB USING utf8)"
            + ",CONVERT(Shop_Items USING utf8),"
            + "CONVERT(Shop_Items_DB USING utf8)"
            + ",CONVERT(Succeeded_buy USING utf8),"
            + "CONVERT(How_much USING utf8) FROM Main_table;");
        results = statement.executeQuery(sql);
        while (results.next()) {
            for (int i = 0; i < s.length; ++i) {
                s[i] = results.getString(i + 1);
            }
            dt.addRow(csvData:s);
        }
        statement.close();
        connection.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

private void btn_exportActionPerformed(java.awt.event.ActionEvent evt) {
    // Ez egy basic export az sql-ből át kell még alakítani a sajátunkra
    String csvFilePath = "Export.csv";
    s = new String[7];
    String line;
    try {
        Connection connection = DriverManager.getConnection(url:jdbcURL, user:username, password);
        String sql = "SELECT * FROM Main_table";
        Statement statement = connection.createStatement();
        results = statement.executeQuery(sql);
        BufferedWriter fileWriter = new BufferedWriter(new FileWriter(csvFilePath));
        while (results.next()) {
            line = "";
            s[0] = results.getInt(columnIndex:1) + "";
            s[1] = results.getString(columnIndex:2);
            s[2] = results.getString(columnIndex:3);
            s[3] = results.getString(columnIndex:4);
            s[4] = results.getString(columnIndex:5);
            s[5] = results.getString(columnIndex:6);
            s[6] = results.getString(columnIndex:7);
            line = String.format("%s[0] + ";" + s[1] + ";" + s[2] + ";" + s[3] + ";" + s[4] + ";" + s[5] + ";" + s[6]);
            fileWriter.write(line);
            fileWriter.newLine();
        }
        statement.close();
        fileWriter.close();
        connection.close();
    } catch (SQLException e) {
        System.out.println("Database error:");
        e.printStackTrace();
    } catch (IOException e) {
        System.out.println("File IO error:");
        e.printStackTrace();
    }
}

```

```

private void btn_findActionPerformed(java.awt.event.ActionEvent evt) {
    // Keresés az adatbázisban és megjeleníteni a táblában
    try {
        Connection connection = DriverManager.getConnection(url:jdbcURL, user:Username, password);
        dt = (DefaultTableModel) table.getModel();
        dt.setRowCount(0);
        Statement statement = connection.createStatement();
        String sql = ("SELECT CONVERT(List_Items USING utf8)"
            + ",CONVERT(List_Items_DB USING utf8)"
            + ",CONVERT(Shop_Items USING utf8)"
            + ",CONVERT(Shop_Items_DB USING utf8)"
            + ",CONVERT(Succeeded_buy USING utf8)"
            + ",CONVERT(How_much USING utf8)"
            + "FROM Main_table "
            + "WHERE List_Items_DB LIKE '" + txt_find.getText() + "' "
            + "OR List_Items LIKE '" + txt_find.getText() + "' "
            + "OR Shop_Items LIKE '" + txt_find.getText() + "' "
            + "OR Shop_Items_DB LIKE '" + txt_find.getText() + "'");
        results = statement.executeQuery(sql);
        while (results.next()) {
            for (int i = 0; i < s.length; ++i) {
                s[i] = results.getString(i + 1);
            }
            dt.addRow(s);
        }
        statement.close();
        connection.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

private void btn_CloseActionPerformed(java.awt.event.ActionEvent evt) {
    // Bezárja az alkalmazást
    WindowEvent closeWindow = new WindowEvent(source:this, id:WindowEvent.WINDOW_CLOSING);
    Toolkit.getDefaultToolkit().getSystemEventQueue().postEvent(closeWindow);
}

```

```

private void btnn_AddRowActionPerformed(java.awt.event.ActionEvent evt) {
    // Hozzáad a táblához egy sort és az adatbázishoz
    s = new String[7];
    try {
        Connection connection = DriverManager.getConnection(url:jdbcURL, user:Username, password);
        Statement statement = connection.createStatement();
        statement.executeUpdate("INSERT INTO Main_table ('List_Items','List_Items_DB','Shop_Items','Shop_Items_DB','Succeeded_buy','How_much') "
            + "VALUES ('" + nothing + "','" + nothing + "','" + nothing + "','" + nothing + "','" + nothing + "','" + nothing + "')");
        dt = (DefaultTableModel) table.getModel();
        dt.addRow(s);
        statement.close();
        connection.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```



```

private void btn_Current_DatabaseActionPerformed(java.awt.event.ActionEvent evt) {
    // Megmutatja az aktuális adatokat az adatbázisban
    dt = (DefaultTableModel) table.getModel();
    dt.setRowCount(0);
    try {
        Connection connection = DriverManager.getConnection(url:jdbcURL, user:username, password);
        String sql = ("SELECT CONVERT(List_Items USING utf8)"
            + ",CONVERT(List_Items_DB USING utf8),"
            + "CONVERT(Shop_Items USING utf8)"
            + ",CONVERT(Shop_Items_DB USING utf8)"
            + ",CONVERT(Succeeded_buy USING utf8)"
            + ",CONVERT(How_much USING utf8) FROM Main_table;");
        statement = connection.createStatement();
        results = statement.executeQuery(sql);
        while (results.next()) {
            for (int i = 0; i < s.length; ++i) {
                s[i] = results.getString(i + 1);
            }
            dt.addRow(s);
        }
        statement.close();
        connection.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

private void btn_DeleteRowActionPerformed(java.awt.event.ActionEvent evt) {
    // Sor törlése táblából és adatbázisból
    try {
        Connection connection = DriverManager.getConnection(url:jdbcURL, user:username, password);
        int index = table.getSelectedRow() + 1;
        int[] rows = new int[table.getRowCount()];
        int i = 0;
        String sql = "Select shop_database.main_table.Rows from shop_database.Main_table;";
        statement = connection.createStatement();
        results = statement.executeQuery(sql);
        while (results.next()) {
            rows[i] = results.getInt(columnIndex:1);
        }
        for (int j = 0; j < rows.length; j++) {
            if (rows[j] == index) {
                sql = ("DELETE FROM shop_database.Main_table WHERE shop_database.main_table.Rows = CAST(" + index + " AS DECIMAL) ;");
                break;
            } else if (rows[j] > index) {
                sql = ("DELETE FROM shop_database.Main_table WHERE shop_database.main_table.Rows > CAST(" + index + " AS DECIMAL) ORDER BY shop_database.main_table.Rows ASC LIMIT 1 ;");
                break;
            }
        }
        statement = connection.createStatement();
        statement.execute(sql);
        dt.removeRow(table.getSelectedRow());
        statement.close();
        connection.close();
    } catch (ArrayIndexOutOfBoundsException boundsException) {
        JOptionPane.showMessageDialog(parentComponent:NULL, message:"Please choose your Row, before usage!", "InfoBox: "
            + "Wrong Usage", messageType:JOptionPane.INFORMATION_MESSAGE);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

private void btn_ResultOfShoppingActionPerformed(java.awt.event.ActionEvent evt) {
    //Meg mondja mit sikerült megvásárolni a listából és ha nem, is sikerült megmondja mennyit sikerült + beírja adatbázisba is
    ArrayList<String> buy_List = new ArrayList<String>();
    ArrayList<String> shop_List = new ArrayList<String>();
    ArrayList<String> result_List = new ArrayList<String>();
    String[] spltStrings_buy = new String[2];
    String[] spltStrings_shop = new String[2];
    int rowindex = 0;
    int listindex = 0;
    try {
        Connection connection = DriverManager.getConnection(url:jdbcURL, user:username, password);
        statement = connection.createStatement();
        String sql = ("SELECT CONVERT(List_Items USING utf8)"
            + ",CONVERT(List_Items_DB USING utf8)"
            + ",CONVERT(Shop_Items USING utf8)"
            + ",CONVERT(Shop_Items_DB USING utf8)"
            + "FROM Main_table ;");
        results = statement.executeQuery(sql);
        while (results.next()) {
            buy_List.add(results.getString(columnIndex:1) + ";" + results.getString(columnIndex:2));
            shop_List.add(results.getString(columnIndex:3) + ";" + results.getString(columnIndex:4));
        }
        for (String buyString : buy_List) {
            if (buyString.equals("objektus:")) {
                spltStrings_buy[0] = "";
                spltStrings_buy[1] = "0";
            } else {
                spltStrings_buy = buyString.split("objektus:");
            }
        }
        for (String shopString : shop_List) {
            if (shopString.equals("objektus:")) {
                spltStrings_shop[0] = "";
                spltStrings_shop[1] = "0";
            } else {
                spltStrings_shop = shopString.split("objektus:");
            }
        }
        if (spltStrings_buy[0].equals(spltStrings_shop[0])) {
            if (Integer.parseInt(spltStrings_buy[1]) <= Integer.parseInt(spltStrings_shop[1])) {
                result_List.add("YES;" + spltStrings_buy[1]);
                table.setValueAt("YES", row:rowindex, column:4);
                table.setValueAt(spltStrings_buy[1], row:rowindex, column:5);
            }
        }
    }
}

```

```

        statement.executeUpdate("UPDATE Main_table SET Succeeded_buy = 'YES',How_much = '" + spltStrings_buy[1] + "' WHERE List_items LIKE '" + spltStrings_buy[0] + "'
        + "ORDER BY shop_database.main_table.Rows ASC LIMIT 1 ;");
    } else if (Integer.parseInt(spltStrings_buy[1]) > Integer.parseInt(spltStrings_shop[1])) {
        result_List.add("NO;" + spltStrings_shop[1]);
        table.setValueAt("NO", row, rowindex, column: 4);
        table.setValueAt(spltStrings_shop[1], row, rowindex, column: 5);
        statement.executeUpdate("UPDATE Main_table SET Succeeded_buy = 'NO',How_much = '" + spltStrings_shop[1] + "' WHERE List_items LIKE '" + spltStrings_buy[0] + "'
        + "ORDER BY shop_database.main_table.Rows ASC LIMIT 1 ;");
    }
    ++rowindex;
}
}
statement.close();
connection.close();
} catch (ArrayIndexOutOfBoundsException boundsException) {
    JOptionPane.showMessageDialog(parentComponent.Null, message: "Please first import your data or Click 'Show Current Data' button, before trying to get the result!", "InfoBox: "
    + "Wrong Usages", messageType: JOptionPane.INFORMATION_MESSAGE);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new MainFrame().setVisible(true);
        }
    });
}
}

```

```

// Variables declaration - do not modify
private javax.swing.JButton btnn_AddRow;
private javax.swing.JButton btnn_Close;
private javax.swing.JButton btnn_Current_Database;
private javax.swing.JButton btnn_DeleteRow;
private javax.swing.JButton btnn_ResultOfShopping;
private javax.swing.JButton btnn_delete;
private javax.swing.JButton btnn_export;
private javax.swing.JButton btnn_find;
private javax.swing.JButton btnn_import;
private javax.swing.JButton btnn_save;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JLabel lbl_Info;
private javax.swing.JTable table;
private javax.swing.JTextField txt_find;
private javax.swing.JTextField txt_input;
// End of variables declaration

```

Nem-funkcionális követelmények:

- A fejlesztés során törekedtünk az egyszerűsége és a könnyű olvashatóságra, így grafikailag és komplexitásában egyszerű szoftverről beszélhetünk

Nyelv és adatbázis kapcsolat:

- A fejlesztés során szembesülnünk kellett a JAVA programozási nyelv sajátosságaival, a NetBeans fejlesztői környezettel és további fejtörést jelentett az MySQL adatbázissal való kapcsolat létrehozása is. Az alkalmazásnak meg kellett teremtenie a kapcsolatot az adatbázissal, amiből a termékek mennyiségét, összehasonlításait és a vásárláshoz kívánt leírásokat kellett importálnunk mind ezt úgy, hogy valós idejű kommunikáció legyen a kettő között. A tervezett alkalmazáshoz UML diagrammot is készítettünk és a hozzá tartozó felhasználói felületeket. A fejlesztés alatt folyamatosan használatban volt a GitHub, amin a projekt különböző fázisait tudtuk feltölteni és nyomon követni. Mindezek mellett a tartalmaznia kellett a most olvasott felhasználói dokumentációt és a bemutató prezentációt a projektről.

Tovább fejlesztési lehetőségek:

- Alkalmazásunk egy prototípus, ami későbbiekben a megrendelővel együtt egyeztetve fejleszthető tovább. Jelenlegi grafikai megjelenítésén tovább színekkel, egyéni gombokkal, animációkkal lehetne kiterjeszteni, ami még esztétikusabb megjelenítést kölcsönözne. Bekerülhetnének további funkciók, mint például egy bejelentkezési menü, amit úgy alakítanánk ki, hogy egyértelműen látszódjon a megrendelő brandje. További funkciók

például valós idejű keresés mikor a keresés mezőben írunk. Kártyás vásárlás lebonyolítását segítő rendszer kialakítása. Bekerülhetne termékek promótálást megjelenítő reklám ablakok. Panel, amin értesülhetnének a vásárlók az esetleges akciós termékekről és még rengeteg más amit készek vagyunk a megrendelővel egyeztetni kívánság szerint, előre.

UML ábra:

