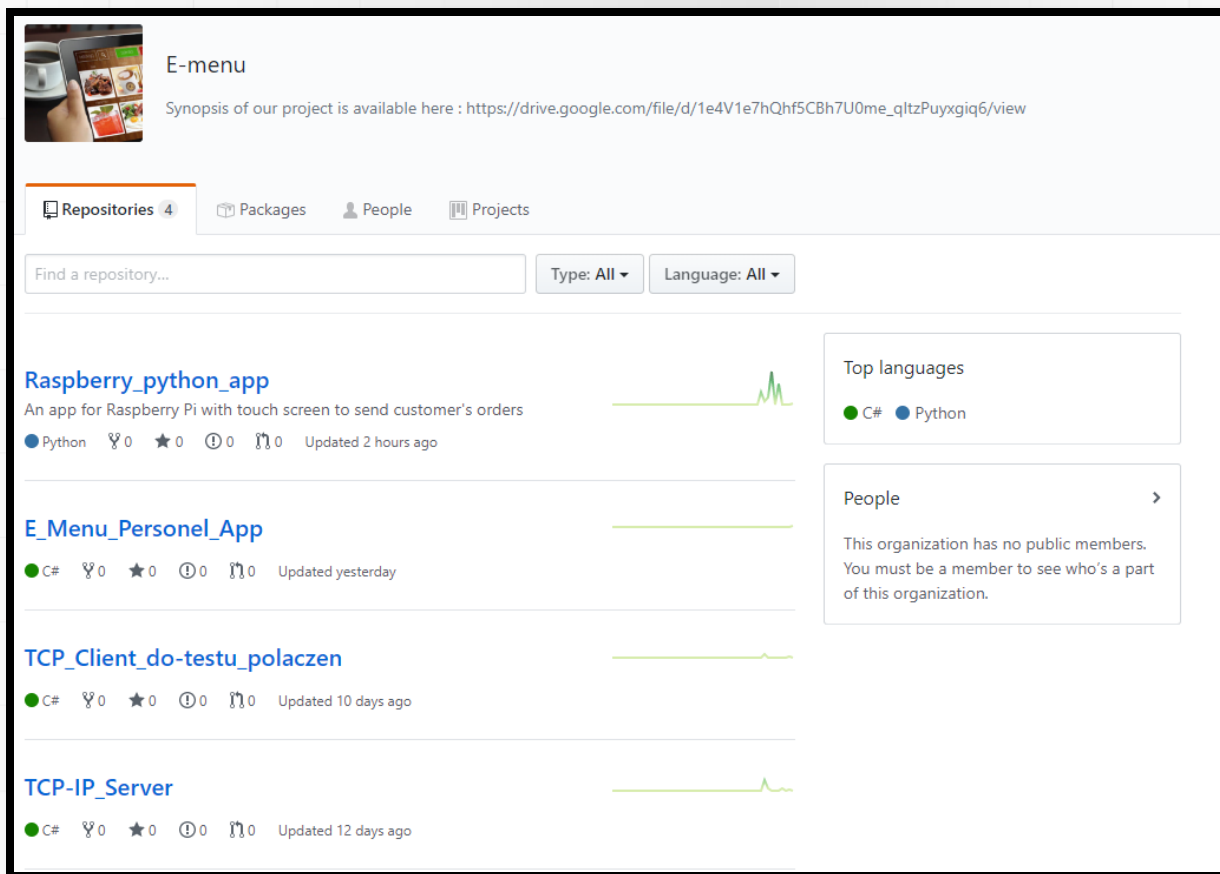




Politechnika
Wrocławska

E-menu

Link do projektu : <https://github.com/E-menu>



Autorzy projektu :

- Przemysław Malec
- Przemysław Widz
- Krzysztof Sitarz



Aplikacja dla klientów



Dane techniczne

Zagadnienie	Opis
Hardware	1. Raspberry Pi 4 model B 2. Ekran dotykowy Waveshare - pojemnościowy 7 " dla Raspberry Pi o rozdzielczości 800 x 480 px
Raspberry Python App (Software)	1. Python 3.7.3 2. Tkinter - GUI (Graphical User Interface) package

Funkcjonalność aplikacji :

- Informacja o dacie na stronie startowej
- Informacja o pogodzie przy użyciu API udostępnionego przez serwis OpenWeather (<https://openweathermap.org/>)

[jako zastąpienie pierwotnego założenia z czujnikami przy stoliku, rozwiązanie to miało być przygotowywane wspólnie, co uniemożliwiła obecna sytuacja]

- Możliwość kolekcjonowania zamówień jako zbiór wybranych przez klienta produktów
- Informacja o poprawności dodania produktu do zamówienia i obecnej cenie zamówienia
- Wyświetlenie podsumowania zamówienia
- Możliwość rezygnacji z zamówienia przez powrót do strony startowej aplikacji
- Możliwość wysłania zamówienia na serwer (do realizacji)
- Informacja o poprawności wysłania zamówienia

Zamówienia

Przechowywane w odpowiednich plikach XML. Przykładowo, w plikach aplikacji, w folderze **menu_items** , znajdują się :

```
pi@raspberrypi:~/Desktop/projekt/Raspberry_python_app/menu_items $ ls -l
total 12
-rw-r--r-- 1 pi pi 1196 May 29 19:32 additives.xml
-rw-r--r-- 1 pi pi 1161 May 29 19:32 drinks.xml
-rw-r--r-- 1 pi pi 1240 May 29 19:32 mainMeals.xml
```

- **additives.xml** - dodatki
- **drinks.xml** - napoje
- **mainMeals.xml** - dania główne

Każdy produkt posiada :

- Nazwę
- Cenę
- Opis
- Wagę



```
<root>

  <dish>
    <title>Pierogi ruskie</title>
    <price>10.00</price>
    <weight>400 g</weight>
    <description>Pyszne pierogi</description>
  </dish>

  <dish>
    <title>Gulasz wolowy</title>
    <price>15.00</price>
    <weight>400 g</weight>
    <description>Pyszny gulasz</description>
  </dish>

  <dish>
    <title>Spaghetti bolognese</title>
    <price>12.00</price>
    <weight>350 g</weight>
    <description>Pyszne spaghetti</description>
  </dish>
```

Kroki działania aplikacji (1)

1) Strona startowa



```
pi@raspberrypi:~/Desktop/projekt/Raspberry_python_app $ ./run.sh
Port is open, connected
```

Kroki działania aplikacji (2)

2) Strony z wyborem produktów

eMenu

Dania główne	Opis	Waga	Cena	Akcja
Pierogi ruskie	Pyszne pierogi	400 g	10.00	<input type="button" value="Dodaj"/>
Gulasz wołowy	Pyszny gulasz	400 g	15.00	<input type="button" value="Dodaj"/>
Spaghetti bolognese	Pyszne spaghetti	350 g	12.00	<input type="button" value="Dodaj"/>
Pstrag pieczony	Pyszny pstrąg	300 g	20.00	<input type="button" value="Dodaj"/>
Filet z kurczaka	Pyszny filet	400 g	11.00	<input type="button" value="Dodaj"/>
Placki ziemniaczane	Pyszne placki	450 g	10.00	<input type="button" value="Dodaj"/>
Bigos	Pyszny bigos	400 g	10.00	<input type="button" value="Dodaj"/>
Kotlet mielony	Pyszny kotlet mielony	300 g	10.00	<input type="button" value="Dodaj"/>

Strona : 1/3

```
Aktualnie zamowione mealsNames : ['Pierogi ruskie']
Aktualnie zamowione mealsPrices : ['10.00']
Aktualny rachunek wynosi : [10.0]
```

```
Aktualnie zamowione mealsNames : ['Pierogi ruskie', 'Frytki']
Aktualnie zamowione mealsPrices : ['10.00', '5.00']
Aktualny rachunek wynosi : [15.0]
```

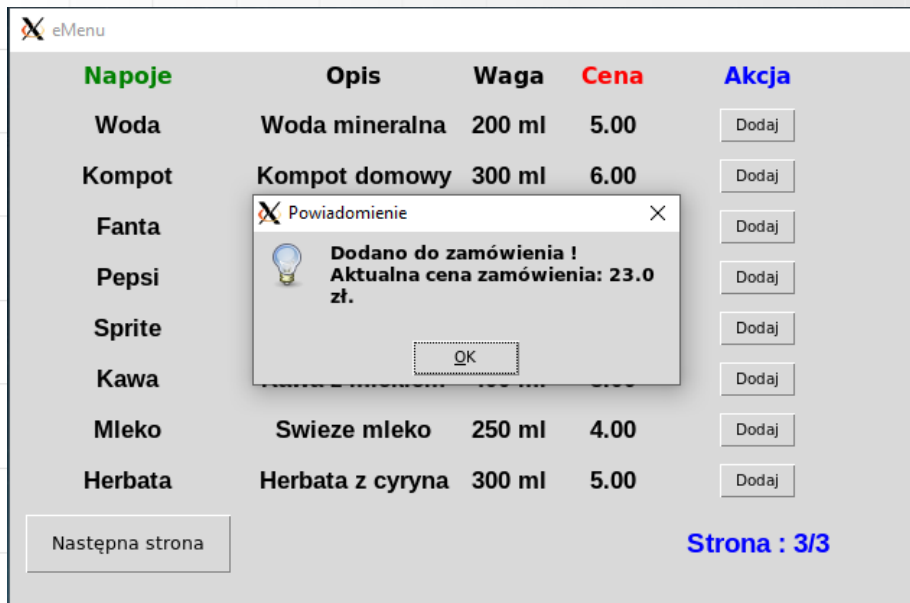
eMenu

Dodatki	Opis	Waga	Cena	Akcja
Frytki	Pyszne frytki	200 g	5.00	<input type="button" value="Dodaj"/>
Salatka	Pyszna салатка	300 g	7.00	<input type="button" value="Dodaj"/>
Krazki cebulowe	Pyszne krazki cebulowe	400 g	10.00	<input type="button" value="Dodaj"/>
Ser panierowany	Pyszny serek	300 g	8.00	<input type="button" value="Dodaj"/>
Makaron	Pyszny makaron	200 g	5.00	<input type="button" value="Dodaj"/>
Ryz	Pyszny ryz	200 g	5.00	<input type="button" value="Dodaj"/>
Ziemniaki	Pyszne ziemniaki	200 g	5.00	<input type="button" value="Dodaj"/>
Buraczki czerwone	Pyszne buraczki	110 g	3.50	<input type="button" value="Dodaj"/>

Strona : 2/3

Kroki działania aplikacji (2) cd.

2) Strony z wyborem produktów




eMenu

Napoje	Opis	Waga	Cena	Akcja
Woda	Woda mineralna	200 ml	5.00	<input type="button" value="Dodaj"/>
Kompot	Kompot domowy	300 ml	6.00	<input type="button" value="Dodaj"/>
Fanta				<input type="button" value="Dodaj"/>
Pepsi				<input type="button" value="Dodaj"/>
Sprite				<input type="button" value="Dodaj"/>
Kawa				<input type="button" value="Dodaj"/>
Mleko	Swieze mleko	250 ml	4.00	<input type="button" value="Dodaj"/>
Herbata	Herbata z cyryna	300 ml	5.00	<input type="button" value="Dodaj"/>

Strona : 3/3

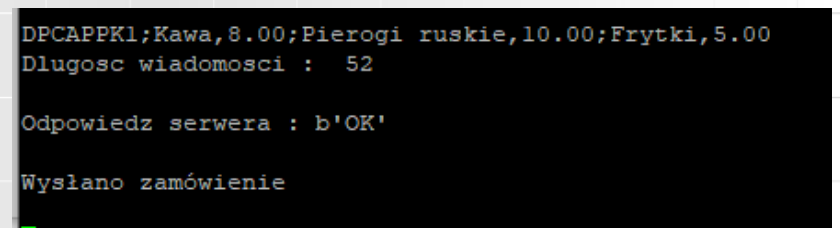
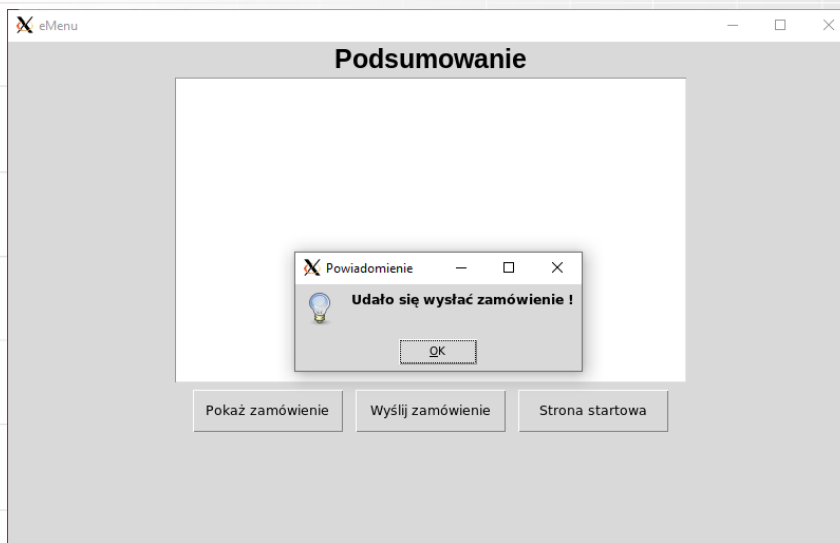
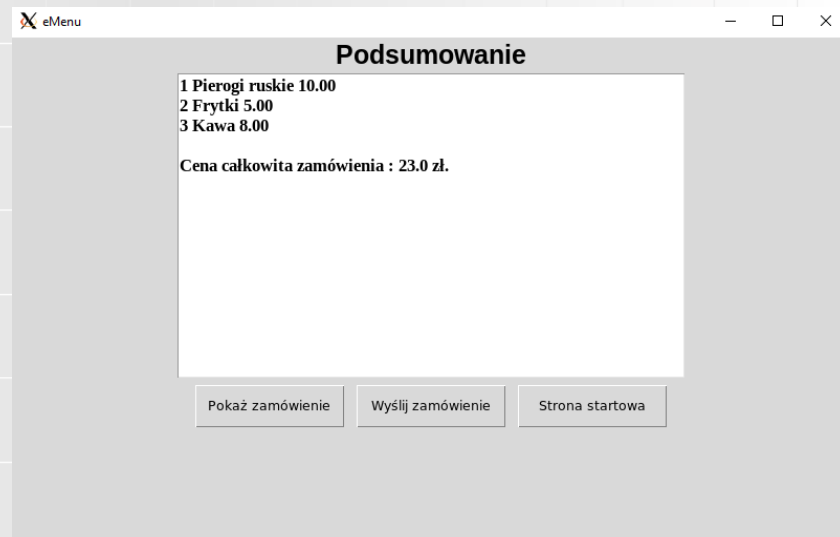
Powiadomienie

 **Dodano do zamówienia !**
Aktualna cena zamówienia: 23.0 zł.

```
Aktualnie zamowione mealsNames : ['Pierogi ruskie', 'Frytki', 'Kawa']  
Aktualnie zamowione mealsPrices : ['10.00', '5.00', '8.00']  
Aktualny rachunek wynosi : [23.0]
```

Kroki działania aplikacji (3)

2) Strona z podsumowaniem oraz opcją wysłania lub rezygnacji z zamówienia



Serwer wielowątkowy

Warstwa transportowa oraz niższe warstwy modelu OSI są realizowane przez system, do ich użycia w aplikacji użyto obiektów typu socket.

Serwer pracuje używając 3 wątków. Pierwszy odpowiada za rejestrację (uruchamia również tymczasowe wątki). Dwa kolejne obsługują odpowiednio, żądania komputerów stacjonarnych lub RPI.

Dane są przechowywane w słowniku którego kluczami są nicki użytkowników z rejestracji a wartością para(socket, oraz stowarzyszony z nim mutex).

Format danych:

|ilość znaków(0-255)|Typ Wiad|ilość odbiorców|Nicki(6znaków, ASCII)|Dane

Jeśli występuje próba rejestracji pod istniejący nick server "pinguje" połączenie z danym użytkownikiem. Jeśli próba nie powiedzie się to znaczy, że połączenie zostało utracone a użytkownik chce podłączyć się jeszcze raz, na co server oczywiście zezwala.

Inicjalizacja wątków obsługi, wątek rejestracji

```

objekt  Kompilowanie  Debugowanie  Test  Analiza  Narzędzia  Rozszerzenia  Okno  Pomoc  Wyszukaj (Ctrl+Q)
Bieżący dokument (UsersData.cs)

UsersData.cs
Server_TCP_IP.UsersData

byte[] data = new byte[i - numberOfrecivers * lengthofNick-1];
Array.ConstrainedCopy(bytes, numberOfrecivers * lengthofNick + 2, data, 1, i - n
data[0] = (byte)(data.Length-1);

return new Packet(reciversarray,data);
}

public void register_desktop(byte[] bytes, TcpClient tcp, int i)
{
    Console.WriteLine( "Register"+tcp.Client.RemoteEndPoint.ToString());
    string Nick = Encoding.ASCII.GetString(bytes, 1, lengthofNick);
    lock(SyncDesktop_users) { // lock on desktop users collection
        if (!Desktop_users.ContainsKey(Nick))
            Desktop_users.Add(Nick, new SyncTCPClient(tcp));
        else {
            try
            {
                lock (Desktop_users[Nick].sync)
                { //ping procedure
                    var stream = Desktop_users[Nick].client.GetStream();
                    Byte[] dummy = new Byte[1];
                    dummy[0] = 0;
                    stream.Write(dummy, 0,1);
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message+ "Registered once again Nick: "+Nick);
                Desktop_users.Remove(Nick);
                Desktop_users.Add(Nick, new SyncTCPClient(tcp)); // Assigned once aga
            }
        }
    }
}

```

```

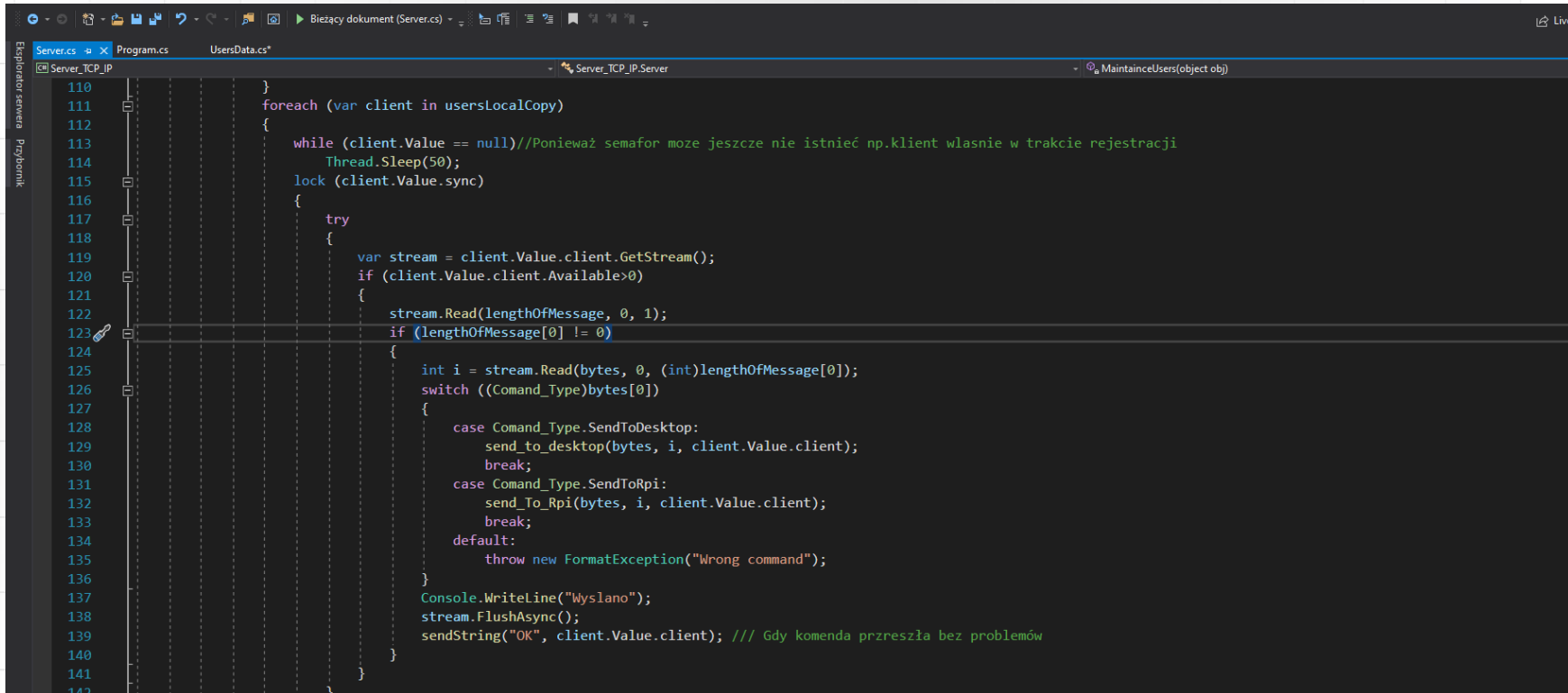
objekt  Kompilowanie  Debugowanie  Test  Analiza  Narzędzia  Rozszerzenia  Okno  Pomoc  Wyszukaj (Ctrl+Q)  TCP-IP_Server
Bieżący dokument (Server.cs)

UsersData.cs*
Server_TCP_IP.Server

public void StartListener()
{
    try
    {
        Maintainer rpi_maintainer = new Maintainer(usersData.Rpi_users, usersData.SyncRpi_users);
        Maintainer desktop_maintainer = new Maintainer(usersData.Desktop_users, usersData.SyncDesk
        Thread MaintainerDesktoptask = new Thread(new ParameterizedThreadStart(MaintainceUsers));
        Thread MaintainerRpitask = new Thread(new ParameterizedThreadStart(MaintainceUsers));
        MaintainerDesktoptask.Start(desktop_maintainer);
        MaintainerRpitask.Start(rpi_maintainer);
        while (true)
        {
            Console.WriteLine("Waiting for a connection...");
            TcpClient client = server.AcceptTcpClient();
            Console.WriteLine("Connected!");
            Thread t = new Thread(new ParameterizedThreadStart(RegisterDevice));
            t.Start(client);
        }
    }
    catch (SocketException e)
    {
        Console.WriteLine("SocketException: {0}", e);
        server.Stop();
    }
}

```

Obsługa użytkowników



```
110 }
111 foreach (var client in usersLocalCopy)
112 {
113     while (client.Value == null) // Ponieważ semafor może jeszcze nie istnieć np. klient właśnie w trakcie rejestracji
114     {
115         Thread.Sleep(50);
116         lock (client.Value.sync)
117         {
118             try
119             {
120                 var stream = client.Value.client.GetStream();
121                 if (client.Value.client.Available > 0)
122                 {
123                     stream.Read(lengthOfMessage, 0, 1);
124                     if (lengthOfMessage[0] != 0)
125                     {
126                         int i = stream.Read(bytes, 0, (int)lengthOfMessage[0]);
127                         switch ((Command_Type)bytes[0])
128                         {
129                             case Command_Type.SendToDesktop:
130                                 send_to_desktop(bytes, i, client.Value.client);
131                                 break;
132                             case Command_Type.SendToRpi:
133                                 send_to_rpi(bytes, i, client.Value.client);
134                                 break;
135                             default:
136                                 throw new FormatException("Wrong command");
137                         }
138                         Console.WriteLine("Wysłano");
139                         stream.FlushAsync();
140                         sendString("OK", client.Value.client); // Gdy komenda przeszła bez problemów
141                     }
142                 }
143             }
144         }
145     }
146 }
```

Przed użyciem socketa uzyskiwany jest do niego dostęp na wyłączność. Jeśli zapis do wskazanego w wiadomości adresata zwróci wyjątek `InvalidOperationException` lub `ObjectDisposedException` to taki klient traktowany jest jako nieaktywny i rozłączany.

Tworzenie pakietów

```
16 public static Packet MakePackettoSend(byte[] bytes,int i)
17 {
18     //pierwszy bajt ilosc adresatow
19     int numberofrecivers = (int)bytes[1];
20     string recivers = Encoding.ASCII.GetString(bytes, 2, numberofrecivers *lengthofNick);
21     string[] reciversarray = recivers.Split('#');
22     byte[] data = new byte[i - numberofrecivers * lengthofNick-1];
23     Array.ConstrainedCopy(bytes, numberofrecivers * lengthofNick + 2, data,1, i - numberofrecivers * lengthofNick - 2);
24     data[0] = (byte)(data.Length-1);
25     return new Packet(reciversarray,data);
26 }
```

```
185 Packet packettosend = UsersData.MakePackettoSend(data, i);
186 foreach (string reciver in packettosend.recivers)
187 {
188     try
189     {
190         if (!usersData.Desktop_users.ContainsKey(reciver))
191             throw new ArgumentException("Desktop_user_list do not contains this nick:" + reciver);
192         lock (usersData.Desktop_users[reciver].sync)
193         {
194             sendPacket(packettosend, usersData.Desktop_users[reciver].client);
195         }
196     }
197 }
198 #region Catch
199 catch (System.IO.IOException e)
200 {
201     Console.WriteLine(e.Message);
202     lock (usersData.SyncDesktop_users) { usersData.Desktop_users.Remove(reciver); }
203     sendString("Lost conenction with " + reciver, sender);
204 }
```



Politechnika
Wrocławska

Aplikacja dla zaplecza gastronomicznego



HR EXCELLENCE IN RESEARCH

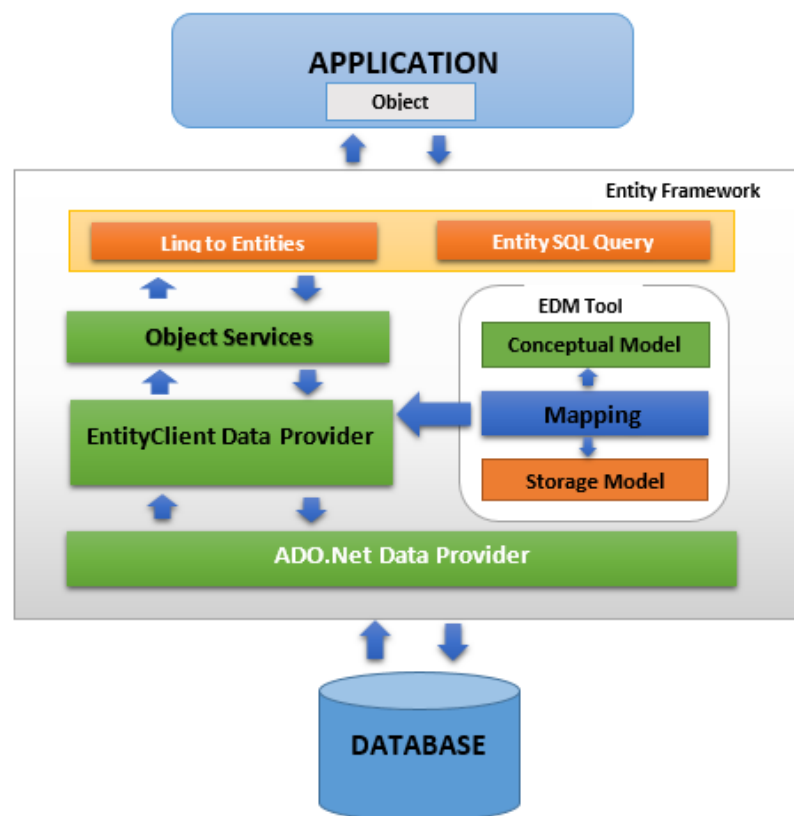
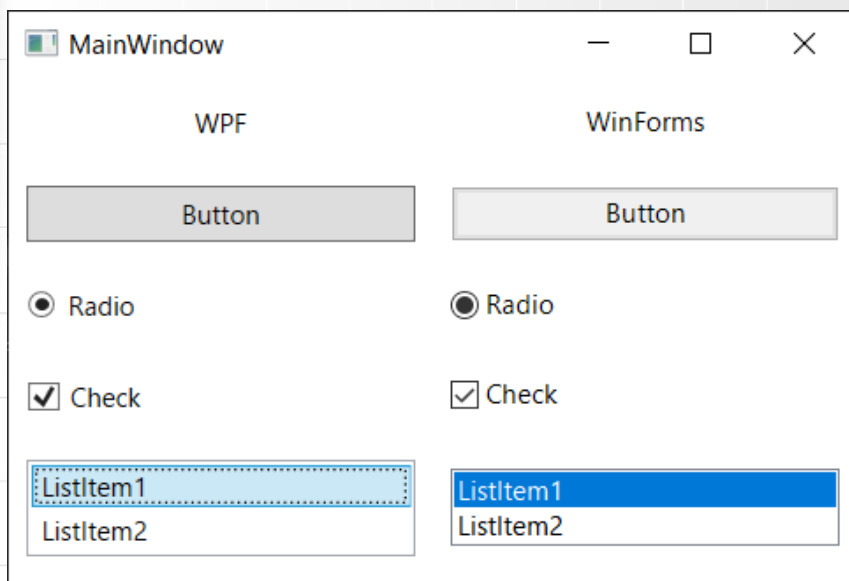
Założenia:



Wykorzystane technologie:

WPF - silnik graficzny

Entity Framework - obiekt do mapowania obiektowo-relacyjnego




Entity Framework


```
public class Order
{
    /// <summary>
    /// Store id of order, primary key
    /// </summary>
    2 references
    public int OrderID { get; set; }





    /// <summary>
    /// Store id of a dish, foreign key
    /// </summary>
    [ForeignKey("Standard")]
    6 references
    public int DishID { set; get; }

    /// <summary>
    /// Store id of a table, foreign key
    /// </summary>
    [ForeignKey("Standard")]
    7 references
    public int TableID { set; get; }

    /// <summary>
    /// Store timeStamp of order
    /// </summary>
    5 references
    public string TimeStamp { set; get; }
}
```

▼  dbo.Orders

▼  Columns

-  OrderID (PK, int, not null)
-  DishID (int, not null)
-  TableID (int, not null)
-  TimeStamp (nvarchar(max), null)

	OrderID	DishID	TableID	TimeStamp
1	128	57	1	18:13:24
2	129	53	1	18:13:24
3	130	58	1	18:13:24
4	131	59	1	18:13:56
5	132	60	1	18:13:56
6	133	52	3	18:15:39
7	134	61	3	18:15:39
8	135	62	3	18:15:39
9	136	60	5	18:16:36
10	137	63	5	18:16:37
11	138	64	5	18:16:37
12	139	57	5	18:16:37

Funkcjonalność



Automatyczne odbieranie zamówień przestanych na serwer



Automatyczne odświeżanie zamówień co 10 sekund



Ręczne dodawanie/usuwanie zamówień




Ręczne odświeżenie zamówień



Politechnika
Wrocławska

Efekt końcowy

23:43:33



Add Dish/Order

Refresh Table

Easy Bistro V2.0

Stolik numer 1				Stolik numer 2				Stolik numer 3			
Nazwa	Cena	Czas zamówienia		Nazwa	Cena	Czas zamówienia		Nazwa	Cena	Czas zamówienia	
Pstrag pieczony	20	18:13:24	Wydano					Herbata	5	18:15:39	Wydano
Woda	5	18:13:24	Wydano					Placki ziemniaczane	10	18:15:39	Wydano
Ziemniaki	5	18:13:24	Wydano								
Fanta	7	18:13:56	Wydano								
Kawa	8	18:13:56	Wydano								
Suma zamówienia: 45				Suma zamówienia: 0				Suma zamówienia: 15			
Stolik numer 4				Stolik numer 5				Stolik numer 6			
Nazwa	Cena	Czas zamówienia		Nazwa	Cena	Czas zamówienia		Nazwa	Cena	Czas zamówienia	
				Kawa	8	18:16:36	Wydano				
				Buraczki czerwone	3.5	18:16:37	Wydano				
				Kotlet mielony	10	18:16:37	Wydano				
				Woda	5	18:16:37	Wydano				
Suma zamówienia: 0				Suma zamówienia: 26.5				Suma zamówienia: 0			

