Politechnika Świętokrzyska

Wydział: ELEKTROTECHNIKI, AUTOMATYKI I INFORMATYKI

Kierunek: Informatyka

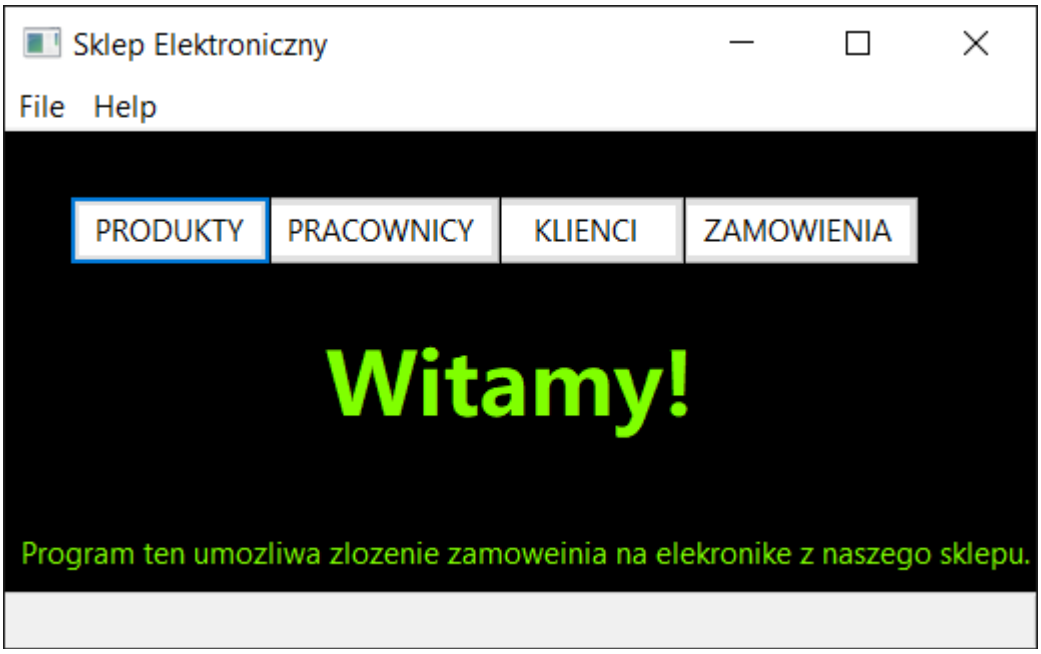
Projekt - Programowanie w języku C2

STUDENT	Patryk Banaś Damian Banasik
Grupa Laboratoryjna	1
GRUPA	2ID11A
PROWADZĄCY	Jacek Wilk-Jakubowski
DATA ODDANIA PROJEKTU	28.01.2020r.
ROK AKADEMICKI	2019/2020

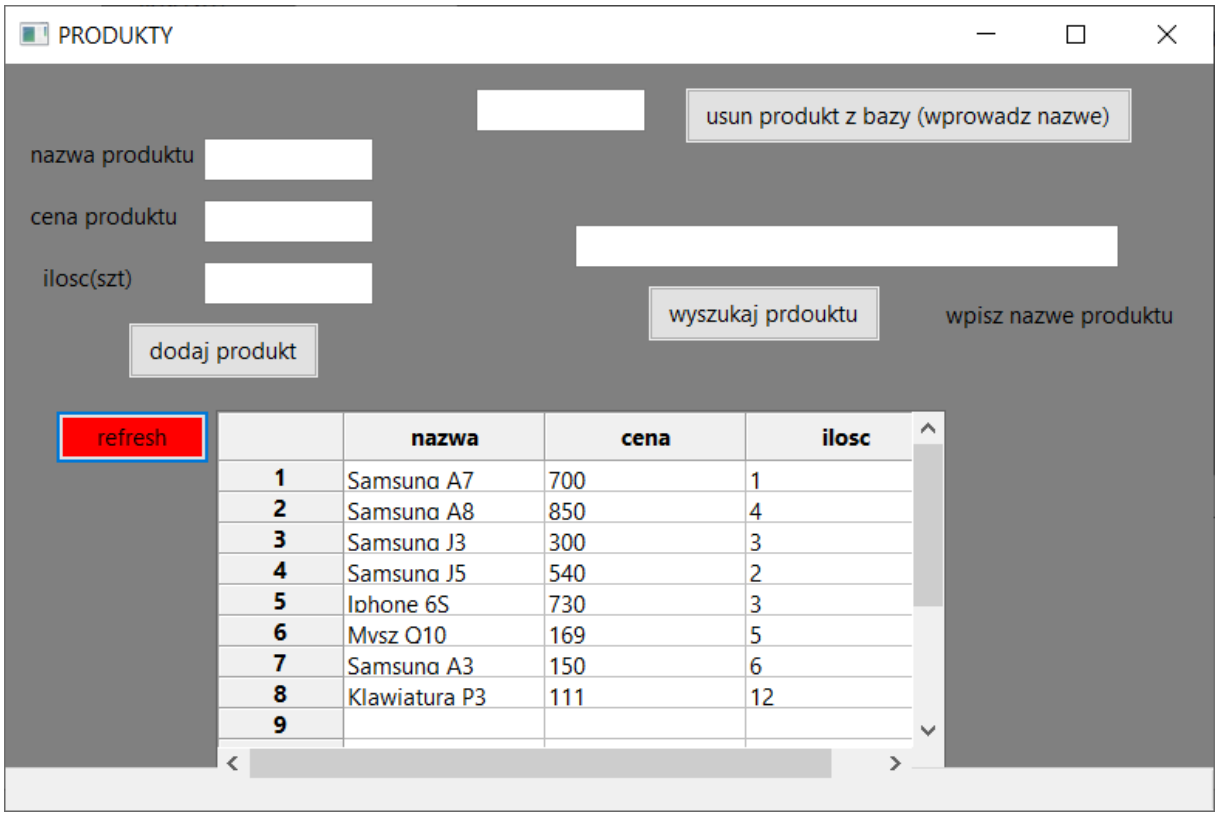
Spis treści:

SKLEP ELEKTRONICZNY

1) Przegląd funkcjonalnośći.

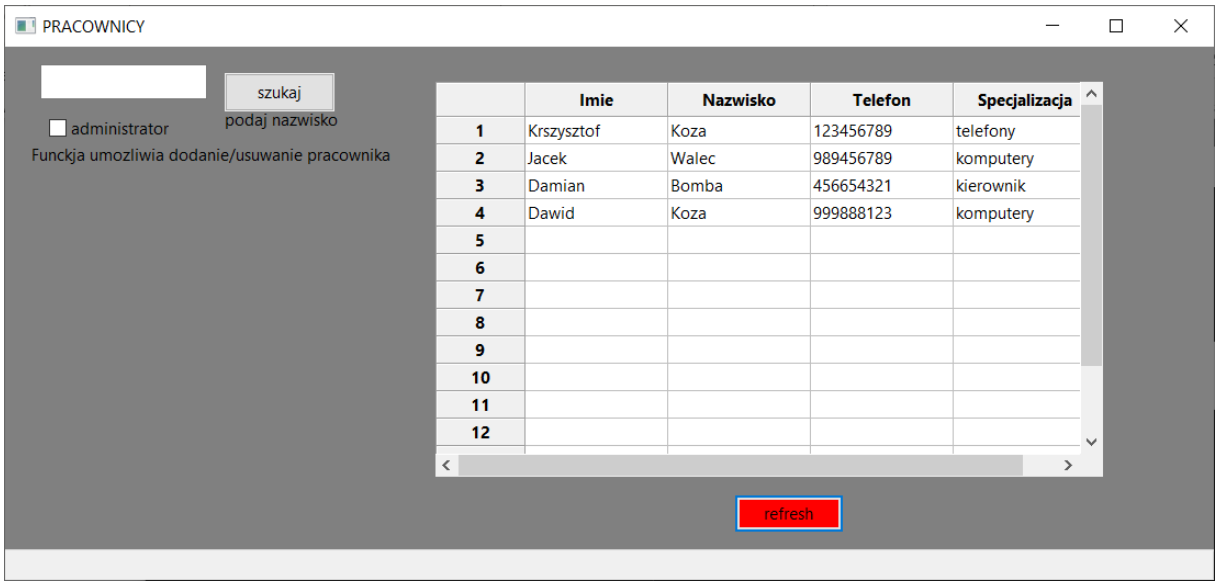


Główne okno naszego sklepu internetowego. Okienko zawiera 2 napisy oraz 4 przyciski, przenoszące użytkownika programu na pod okienka umożliwiające pracę z bazą danych SQLite3.

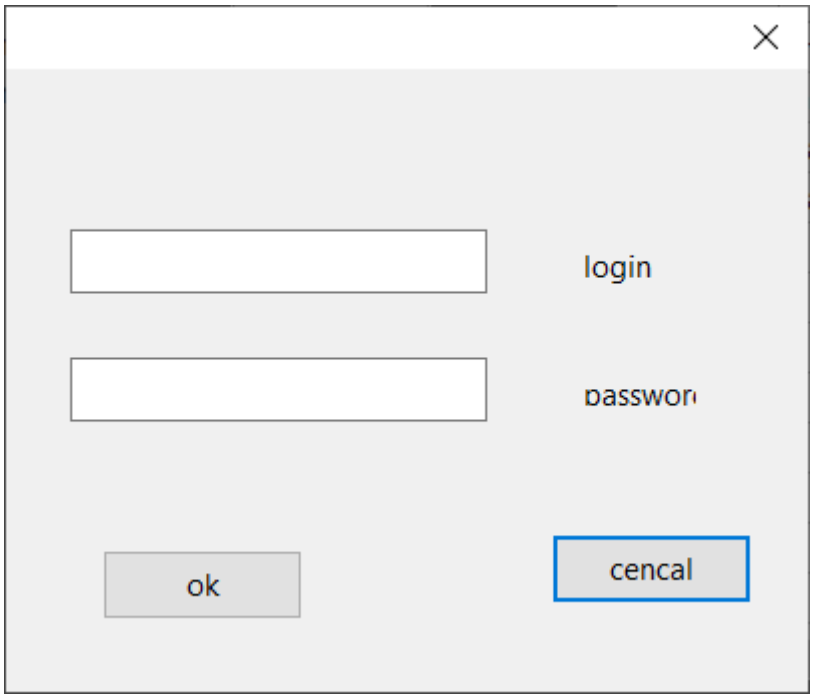


Pod przyciskiem „PRODUKTY” kryje się tabela umożlwiająca, przeglądanie zawartości magazynu sklepowego. Oraz okienka pozwalające na dodanie, usuwanie i wyszukiwanie produktów.

Czerwony przycisk „refresh” służy uzupełnienia danymi tabeli. Przyciski automatycznie po kliknięciu odświeżają zawartość tabeli.



Po kliknięciu przycisku „PRACOWNICY” wyświetla nam się okno umożliwiające przegląd bazy zawierającej pracowników naszego sklepu elektronicznego. Mamy również tutaj „CheckBox” który po kliknięciu, umożliwia dalsze funkcje komunikacji z bazą danych, takich jak dodawanie i usuwanie pracownika. Funkcje te zabezpieczone są okienkiem logowania, po to aby nikt nie powołany nie namieszał nam w naszych danych.

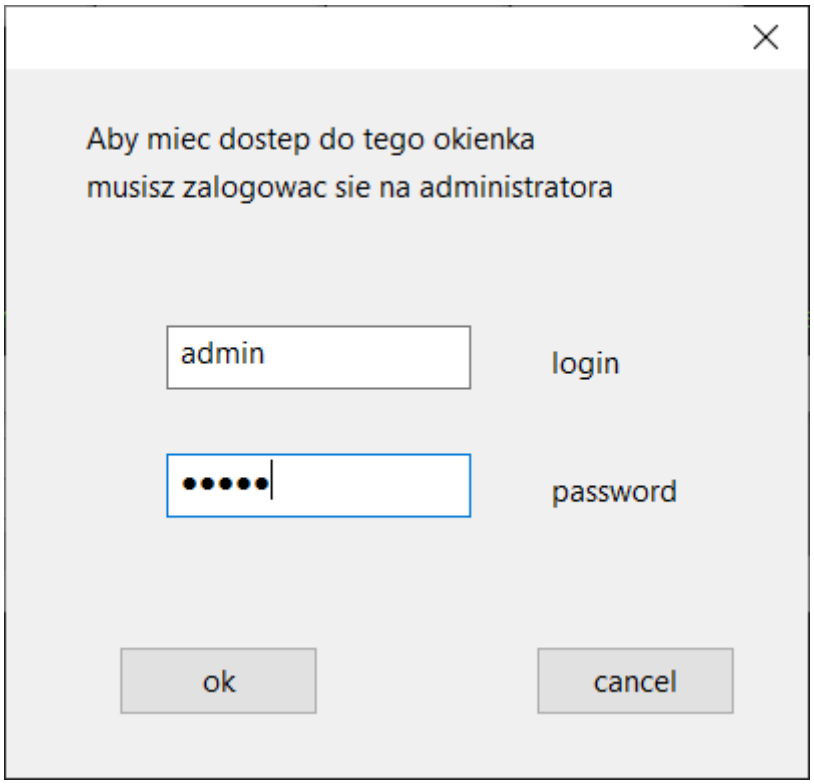


Dane do logowania to

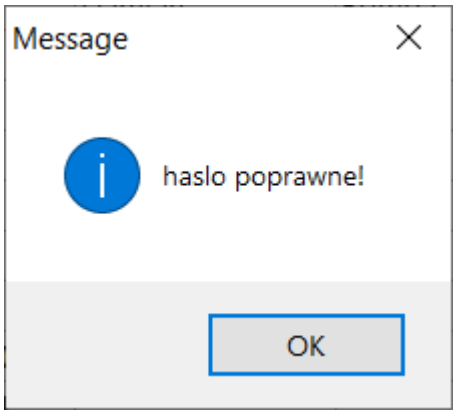
Login: admin

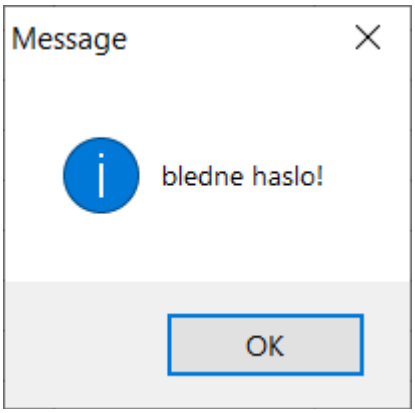
Password: admin

Po poprawnym wprowadzeniu hasła zostaniemy poinformowani przez okienko dialogowe o jego poprawności, analogicznie w przeciwnym wypadku o błędnym haśle.

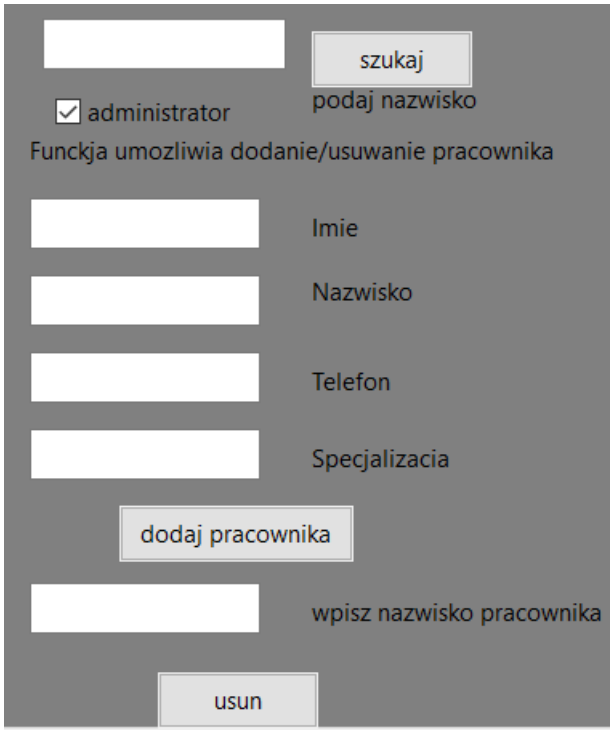


Podgląd hasła jest zaszyfrowany.



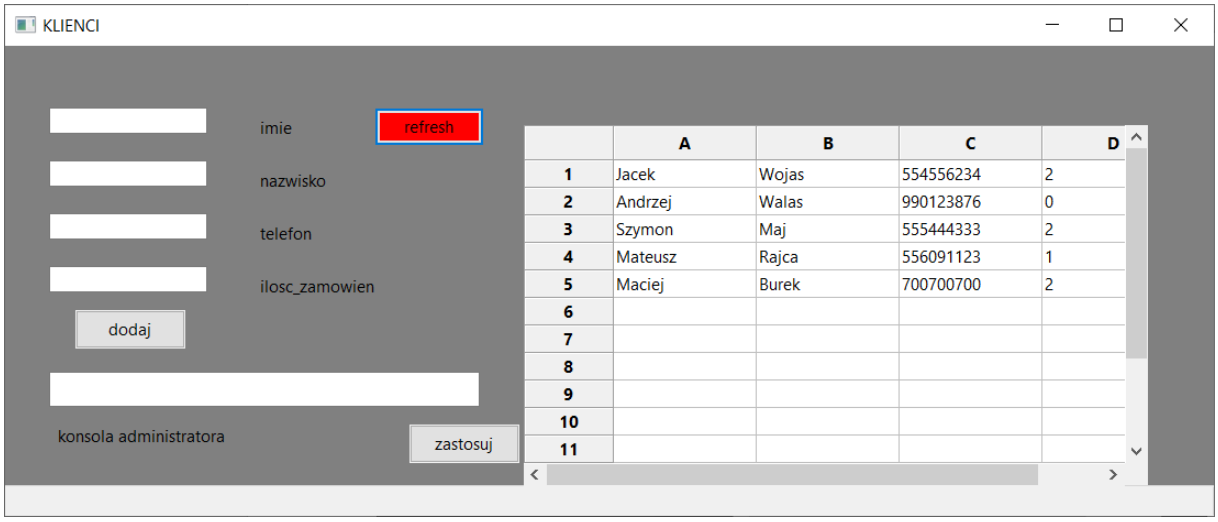


Po zalogowaniu na konto administratorskie naszym oczom pojawiają się, takie dodatkowe możliwości:



Następny przycisk „KLIENCI” jest zabezpieczony hasłem administratora, aby nie udostępniać osobą 3 danych poufnych naszych klientów.

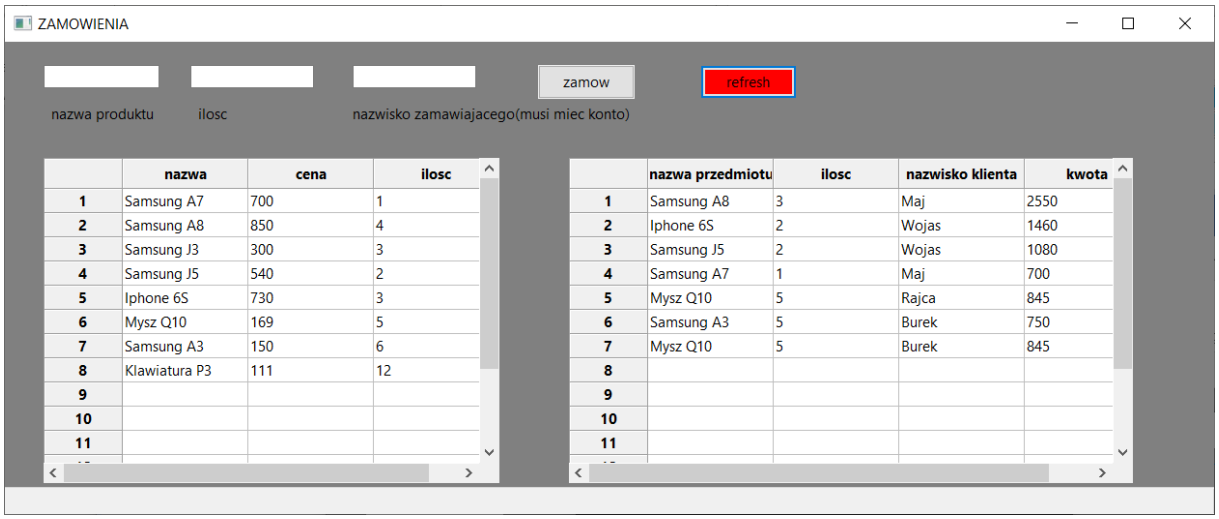
Hasło jest takie samo jak wcześniej, oraz również program wyświetli nam komunikaty o poprawności hasła.



W tym okienku mamy możliwość dodawania nowych nabywców naszych towarów. Oraz konsole administratorską, dzięki której możemy zarządzać całą bazą, wprowadzając dostępne komendy (SQLite3).

Po naciśnięciu przycisku dodaj wyświetlają nam się napisy pomocnicze.

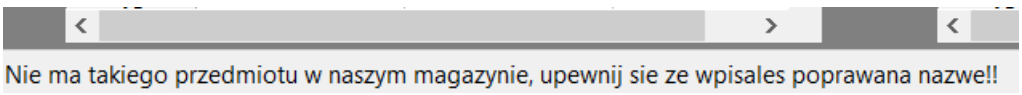
Po kliknięciu przycisku „ZAMOWIENIA” ukazuje nam się takie oto okno.

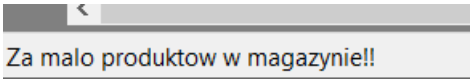


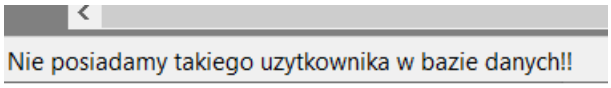
	nazwa	cena	ilosc
1	Samsung A7	700	1
2	Samsung A8	850	4
3	Samsung J3	300	3
4	Samsung J5	540	2
5	Iphone 6S	730	3
6	Mysz Q10	169	5
7	Samsung A3	150	6
8	Klawiatura P3	111	12
9			
10			
11			

	nazwa przedmiotu	ilosc	nazwisko klienta	kwota
1	Samsung A8	3	Maj	2550
2	Iphone 6S	2	Wojas	1460
3	Samsung J5	2	Wojas	1080
4	Samsung A7	1	Maj	700
5	Mysz Q10	5	Rajca	845
6	Samsung A3	5	Burek	750
7	Mysz Q10	5	Burek	845
8				
9				
10				
11				

W którym mamy mamy możliwość podglądu zawartości naszego magazynu oraz listy aktywnych zamówień, zwartych w osobnych tabelach. Oraz standardowo przycisk refresh. Dodawanie nowych zamówień jest zabezpieczone, przed wprowadzaniem nieprawidłowych danych. Między innymi złą nazwą przedmiotu, nie wystarczającą liczbą produktów w sklepie oraz brakiem nazwiska w naszej bazie danych.



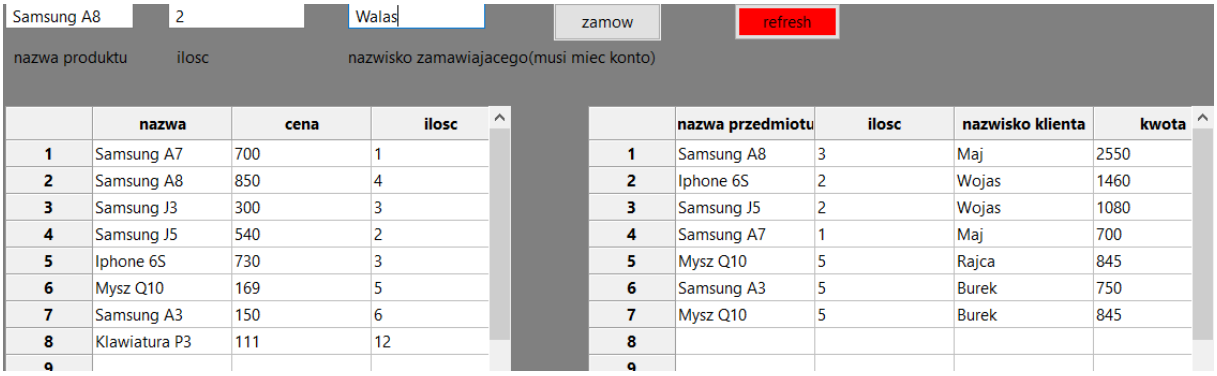




Komunikaty te podawane są w „StatusBar” na dole okienka.

	Andrzej	Walas	990123876	0
	Szymon	Maj	555444333	2
	Mateusz	Rajca	556091123	1

Teraz trochę o tym jak działa w naszym sklepie baza. Pan Andrzej Walas ma 0 zamówień.



Wprowadzamy wszystkie potrzebne dane. Klikamy przycisk „zamów”.

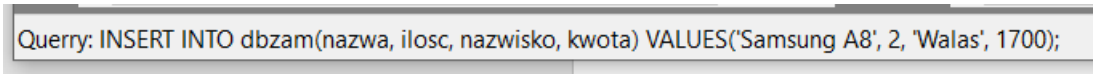
	nazwa	cena	ilosc
1	Samsung A7	700	1
2	Samsung A8	850	2
3	Samsung J3	300	3
4	Samsung J5	540	2
5	Iphone 6S	730	3
6	Mysz Q10	169	5
7	Samsung A3	150	6
8	Klawiatura P3	111	12
9			

	nazwa przedmiotu	ilosc	nazwisko klienta	kwota
1	Samsung A8	3	Maj	2550
2	Iphone 6S	2	Wojas	1460
3	Samsung J5	2	Wojas	1080
4	Samsung A7	1	Maj	700
5	Mysz Q10	5	Rajca	845
6	Samsung A3	5	Burek	750
7	Mysz Q10	5	Burek	845
8	Samsung A8	2	Walas	1700
9				

Ilość dostępnych produktów o podanej nazwie zmniejsza nam się o podaną ilość. A w Tabeli zamówienia pojawia nam się nowe zamówienie o podanych danych, już z sumowaną kwotą do zapłaty. Dane o cenie produktu pobierane są z tabeli produkty i mnożone przez ilość zamówionych przez nas przedmiotów.

	A	B	C	D
1	Jacek	Wojas	554556234	2
2	Andrzej	Walas	990123876	1
3	Szymon	Maj	555444333	2
4	Mateusz	Rajca	556091123	1
5	Maciej	Burek	700700700	2
6				

W tabeli klienci natomiast do liczby zamówień została dodana jedynka. Oraz w każdym okienku, gdy modernizujemy naszą bazę, w „StatusBar” Wyświetla nam się polecenie które aktualnie wykonujemy.



2) Kod programu

2.1 Główne okno.

```
#include "sklep_elektronicznyMain.h"
#include <wx/msgdlg.h>
#include "pro.h"
#include "pra.h"
#include "kli.h"
#include "zam.h"
#include "logowanie2.h"
```

Zaimplementowanie potrzebnych plików nagłówkowych.

Odwołanie do pliku wx/msgdlg.h jest potrzebne w naszym przypadku do okienek „MessageBox”

```
139  void sklep_elektronicznyFrame::OnButton1Click(wxCommandEvent& event)
140  {
141      pro* frm = new pro(this);
142      frm->Show();
143  }
144
145  void sklep_elektronicznyFrame::OnButton2Click(wxCommandEvent& event)
146  {
147      pra* frm = new pra(this);
148      frm->Show();
149  }
150
151
152  void sklep_elektronicznyFrame::OnButton3Click(wxCommandEvent& event)
153  {
154
155      wxString a,b;
156
157      logowanie2 okienko2(this);
158      int x=okienko2.ShowModal();
159      if(x=wxID_OK)
160      {
161          a=okienko2.Func2("log2");
162          b=okienko2.Func2("pas2");
163          if(a=="admin"&&b=="admin")
164          {
165              wxMessageBox( wxT("haslo poprawne!") );
166              kli* frm = new kli(this);
167              frm->Show();
168
169          } else wxMessageBox( wxT("bledne haslo!") );
170
171  }
172  }
173
174  void sklep_elektronicznyFrame::OnButton4Click(wxCommandEvent& event)
175  {
176      zam* frm = new zam(this);
177      frm->Show();
```

Linijki 141-142 odpowiedzialne są za wyświetlenie okienka PRODUKTY po kliknięciu przycisku.

Linijki 147-148 odpowiedzialne są za wyświetlenie okienka PRACOWNICY po kliknięciu przycisku.

Linijki 176-177 odpowiedzialne są za wyświetlenie okienka ZAMOWIENIA po kliknięciu przycisku.

Natomiast mechanizm działający po kliknięciu przycisku o nazwie „Button3” jest trochę bardziej skomplikowane

Deklarujemy 2 ciągi znaków a i b.

Następnie wyświetla się okno dialogowe, z dwoma przyciskami. Okienko logowania używa funkcji którą trzeba stworzyć w pliku nagłówkowym logowanie2.h

```
    private:

        //(*Handlers(logowanie2)
        void OnButton1Click(wxCommandEvent& event);
        void OnButton2Click(wxCommandEvent& event);
        //*)

        public:

         wxString Func2(wxString a)
        {
            wxString b;
            if(a=="log2")b=log2->GetValue();
            if(a=="pas2")b=pas2->GetValue();
            return b;
        }

        DECLARE_EVENT_TABLE()
};
```

która wygląda w taki sposób.

Pobiera ona 2 wyrazy i porównuje z wyrazami które umożliwiają dostęp do naszego okienka.

Jeżeli hasło jest poprawne to wyświetla nam się komunikat o poprawnym haśle oraz wyświetla nam się nowe okno. W przeciwnym wypadku ostatnim krokiem działania funkcji jest wyświetlenie komunikatu o błędnym haśle.

2.2 Okienko Produkty

W pliku pro.cpp musimy zainicjować nową bazę

```
SetStatusBar(StatusBar1);
Center();

Connect(ID_BUTTON1,wxEVT_COMMAND_BUTTON_CLICKED,(wxO
Connect(ID_TEXTCTRL4,wxEVT_COMMAND_TEXT_UPDATED,(wxO
Connect(ID_BUTTON2,wxEVT_COMMAND_BUTTON_CLICKED,(wxO
Connect(ID_TEXTCTRL5,wxEVT_COMMAND_TEXT_UPDATED,(wxO
Connect(ID_BUTTON3,wxEVT_COMMAND_BUTTON_CLICKED,(wxO
Connect(ID_BUTTON4,wxEVT_COMMAND_BUTTON_CLICKED,(wxO
//*)

db = new wxSQLite3Database();
```

Po kliknieciu tego przycisku który ma nazwę „Button1”

```
void pro::OnButton1Click(wxCommandEvent& event)
{
    wxString a,b,c,t;
    a=TextCtrl1->GetValue();
    b=TextCtrl2->GetValue();
    c=TextCtrl3->GetValue();
    t="INSERT INTO dbpro (nazwa, cena, ilosc) VALUES ('"+a+"', "+b+", "+c+");";
    wxString testDBName = wxGetCwd() + wxT("/database.db");
    db->Open(testDBName);
    db->ExecuteUpdate(wxString::Format(_("%s"),t));
    db->Close();
    StatusBar1->SetStatusText(_("Querry: ")+t);
    TextCtrl1->SetValue(_(""));
    TextCtrl2->SetValue(_(""));
    TextCtrl3->SetValue(_(""));
    OnButton4Click(event);
}
```

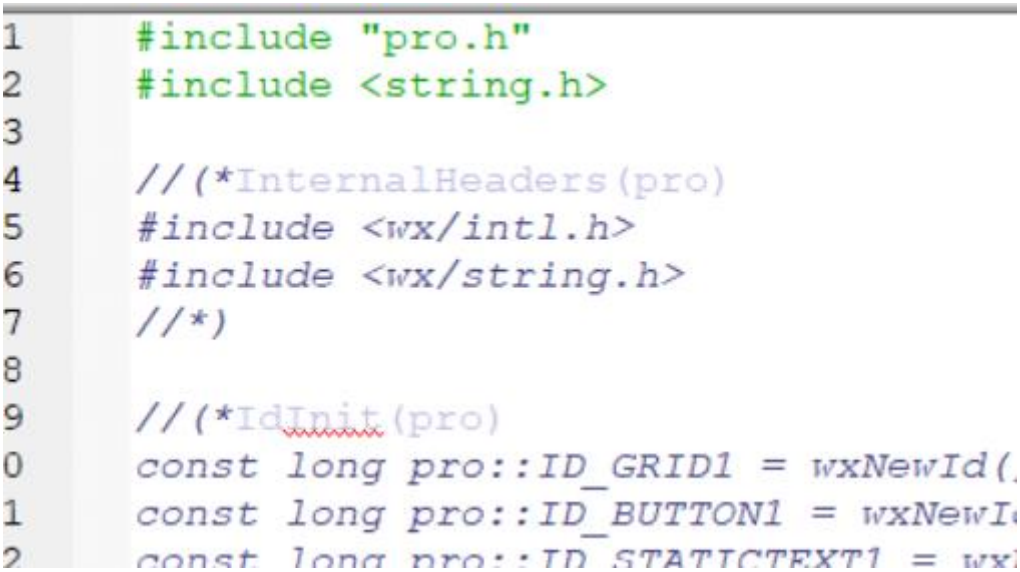
Program zadziała w taki sposób jak opisuje powyższy kod.

Czyli tworzymy 4 zmienne w których zapiszemy wpisane przez nas dane. TextCtrl1, TextCtrl1, TextCtrl1 to nazwa naszych pół do wpisywania danych.

Są ustawione w odpowiedniej kolejnośći 1->2->3.

Przez Funkcję GetValue() dane są pobierane z pól i zapisywane do naszych już wcześniej zadeklarowanych zmiennych.

W zmiennej „t” będzie zapisane polecenie działające na bazie danych dzięki wcześniej zadeklarowanej bibliotece string



Możliwe są operacje na łańcuchach znaków takie jak np.: dodawanie dwóch ciągów. Gotowe już polecenie zapisujemy w zmiennej.

Następnie łączymy się z naszą bazą danych której plik znajduje się w głównym folderze pod nazwą „database.db”

Gotowymi poleceniami otwieramy naszą bazę i ją aktualizujemy, wprowadzając następne polecenie.

Po przeprowadzanym działaniu baza jest zamykana poleceniem

db->Close() .

W „StatusBar” znajdującym się na dole okienka wyświetli nam się zastosowane przez nas polecenie jeśli się powiedzie jego wykonanie.

Następnie usunie się zawartość okienek, aby przy ponownym wprowadzaniu zawartości nie trzeba ich czyścić.

Ostatecznie Uruchomi się funkcja mająca na celu odświeżenie zawartości tabelki.

Która działa w taki sposób:

```
155
156 void pro::OnButton4Click(wxCommandEvent& event)
157 {
158     db = new wxSQLite3Database();
159
160     wxString testDBName = wxGetCwd() + wxT("/database.db");
161     db->Open(testDBName);
162     wxSQLite3ResultSet Res = db->ExecuteQuery(wxString::Format(_("%s"),"SELECT * FROM dbpro;"));
163     Grid1->ClearGrid();
164     int count = 0;
165
166     while (Res.NextRow())
167     {
168         Grid1->SetCellValue(count,0,wxString::Format(_("%s"),Res.GetAsString(0)) ) ;
169         Grid1->SetCellValue(count,1,wxString::Format(_("%d"),Res.GetInt(1)) ) ;
170         Grid1->SetCellValue(count,2,wxString::Format(_("%d"),Res.GetInt(2)) ) ;
171         count++;
172     }
173     Res.Finalize();
174     db->Close();
175 }
```

Baza danych jest otwierana, przez odpowiednie sztywno ustalone polecenie wybieramy cała zawartość tabeli o nazwie „dbpro”

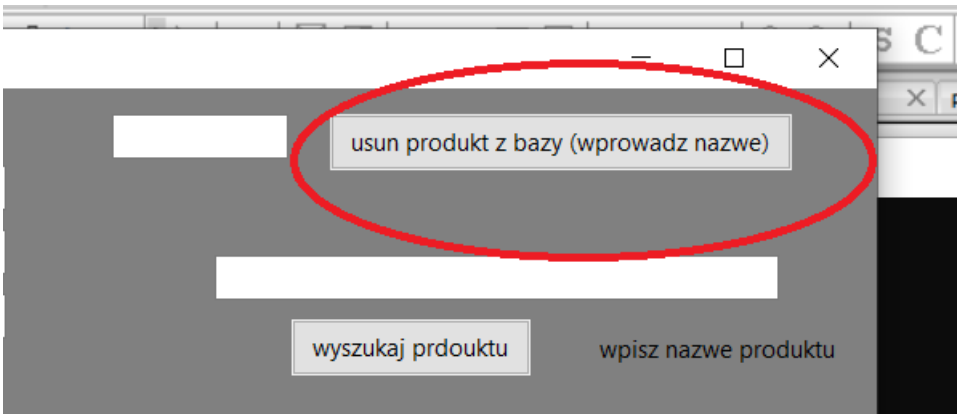
Usuwamy dotychczasową zawartość naszej tabelki.

Następnie poprzez pętle while będziemy uzupełniać rekordy naszej tabeli dopóki następny rekord pobierany z bazy danych nie będzie pusty.

„Grid1” to nazwa naszej tabeli ustawiamy w niej format treści jaki ma być wstawiony oraz numeracie pobieranych okienek analogicznie od 0-2; ponieważ mamy 3 kolumny: nazwa, cena, ilość.

Ostatecznie zamykamy połączenie z bazą danych.

Po naciśnięciu tego przycisku:



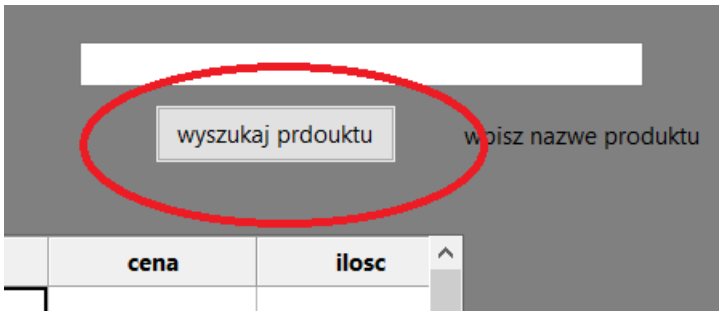
Będzie wykonywał się następujący kod programu:

```
138  void pro::OnButton2Click(wxCommandEvent& event)
139  {
140      wxString a,t;
141      a=TextCtrl4->GetValue();
142      t="DELETE FROM dbpro WHERE nazwa='"+a+"';";
143      wxString testDBName = wxGetCwd() + wxT("/database.db");
144      db->Open(testDBName);
145      db->ExecuteUpdate(wxString::Format(_("%s"),t));
146      db->Close();
147      StatusBar1->SetStatusText(_("Querry: ")+t);
148      TextCtrl4->SetValue(_(""));
149      OnButton4Click(event);
150  }
```

Działa on w następujący sposób:

Inicjujemy dwie zmienne, pobieramy zapisaną wartość w naszym kolejnym „TextCtrl4” i zapisujemy ją do zmiennej a.

W zmiennej t zapisujemy polecenie usuwający rekord o podanej nazwie. Łączymy się z bazą, aktualizujemy tabele podanym poleceniem następnie ją zamykamy, wyświetlamy treść naszego polecenia, usuwamy wartość z okienka i analogicznie odświeżamy zawartość tabeli.



Ostatni przycisk „wyszukaj produktu” w tym okienku uruchamia poniższy kod:

```
110
111 void pro::OnButton3Click(wxCommandEvent& event)
112 {
113     wxString a,t;
114     a=TextCtrl5->GetValue();
115     t="SELECT * FROM dbpro WHERE nazwa='"+a+"';";
116     wxString testDBName = wxGetCwd() + wxT("/database.db");
117     db->Open(testDBName);
118     wxSQLite3ResultSet Res = db->ExecuteQuery(wxString::Format(_("%s"),t));
119     Grid1->ClearGrid();
120     int count = 0;
121
122     while (Res.NextRow())
123     {
124         Grid1->SetCellValue(count,0,wxString::Format(_("%s"),Res.GetAsString(0)) ) ;
125         Grid1->SetCellValue(count,1,wxString::Format(_("%d"),Res.GetInt(1)) ) ;
126         Grid1->SetCellValue(count,2,wxString::Format(_("%d"),Res.GetInt(2)) ) ;
127         count++;
128     }
129     Res.Finalize();
130     db->Close();
131
132 }
```

Tworzy 2 zmienne typu string, pobiera wartość zapisuje, pobrana wartość wraz z poleceniem do zmiennej „t” łaczy się z bazą danych otwiera ją wprowadza przygotowane polecenie, czyści naszą tabelę z danych i wprowadza informacje analogicznie jak w przypadku funkcji: OnButton4Click().

Następnie zamyka bazę.

Wszędzie tam gdzie będziemy pracować z baza musimy dodać taki kod w pliku .cpp

```
db = new wxSQLite3Database();
```

Oraz w pliku .h taki:

```
81          void OnButton2Click1(wxCommandEvent& ev
82          void OnButton3Click(wxCommandEvent& eve
83          void OnButton4Click(wxCommandEvent& eve
84          //*)
85
86          wxSQLite3Database* db;
87
88          DECLARE_EVENT_TABLE()
89  };
90
```

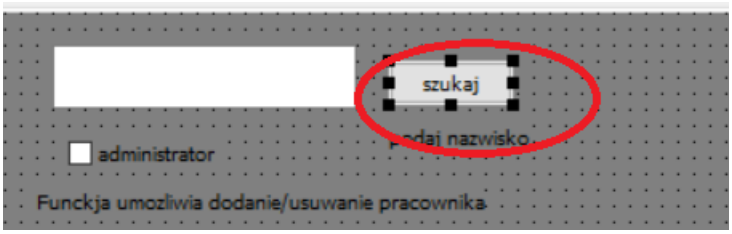
Do tego należy w pliku .h dodać takie biblioteki:

```
11  #include <wx/textctrl.h>
12  //*)
13
14  #include <wx/msgdlg.h>
15  #include "wx/wxsqlite3.h"
16
17  class pra: public wxFrame
18  {
19      public:
```

Aby praca z bazą danych była możliwa.

2.3 Okienko Pracownicy

W pliku pra.cpp po kliknięciu tego przycisku:



Zadziała taka funkcja:

```
181  void pra::OnButton2Click1(wxCommandEvent& event)
182  {
183      wxString a,t;
184      a=TextCtrl5->GetValue();
185      t="SELECT * FROM dbpracow WHERE nazwisko='"+a+"';";
186      wxString testDBName = wxGetCwd() + wxT("/database.db");
187      db->Open(testDBName);
188      wxSQLite3ResultSet Res = db->ExecuteQuery(wxString::Format(_("%s"),t));
189      Grid1->ClearGrid();
190      int count = 0;
191
192      while (Res.NextRow())
193      {
194          Grid1->SetCellValue(count,0,wxString::Format(_("%s"),Res.GetAsString(0)) ) ;
195          Grid1->SetCellValue(count,1,wxString::Format(_("%s"),Res.GetAsString(1)) ) ;
196          Grid1->SetCellValue(count,2,wxString::Format(_("%d"),Res.GetInt(2)) ) ;
197          Grid1->SetCellValue(count,3,wxString::Format(_("%s"),Res.GetAsString(3)) ) ;
198          count++;
199      }
200      TextCtrl4->SetValue(_(""));
201      Res.Finalize();
202      db->Close();
203  }
204
```

Która działa w identyczny sposób jak funkcja wyszukująca w tabeli dbpro(produkty).

Po kliknięciu CheckBoxa Administraotr wyświetli nam się okienko logowania.

```
118 void pra::OnCheckBox1Click(wxCommandEvent& event)
119 {
120     wxString a,b;
121
122     logowanie okienko(this);
123     int x=okienko.ShowModal();
124     if(x=wxID_OK)
125     {
126         a=okienko.Funcl("log");
127         b=okienko.Funcl("pas");
128         if(a=="admin"&&b=="admin")
129         {
130             TextCtrl1->Show(true);
131             TextCtrl2->Show(true);
132             TextCtrl3->Show(true);
133             TextCtrl4->Show(true);
134             TextCtrl6->Show(true);
135             StaticText1->Show(true);
136             StaticText2->Show(true);
137             StaticText3->Show(true);
138             StaticText4->Show(true);
139             StaticText6->Show(true);
140             Button1->Show(true);
141             Button3->Show(true);
142             wxMessageBox( wxT("haslo poprawne!") );
143         } else wxMessageBox( wxT("bledne haslo!") );
144
145     }
146
147
148
149
```

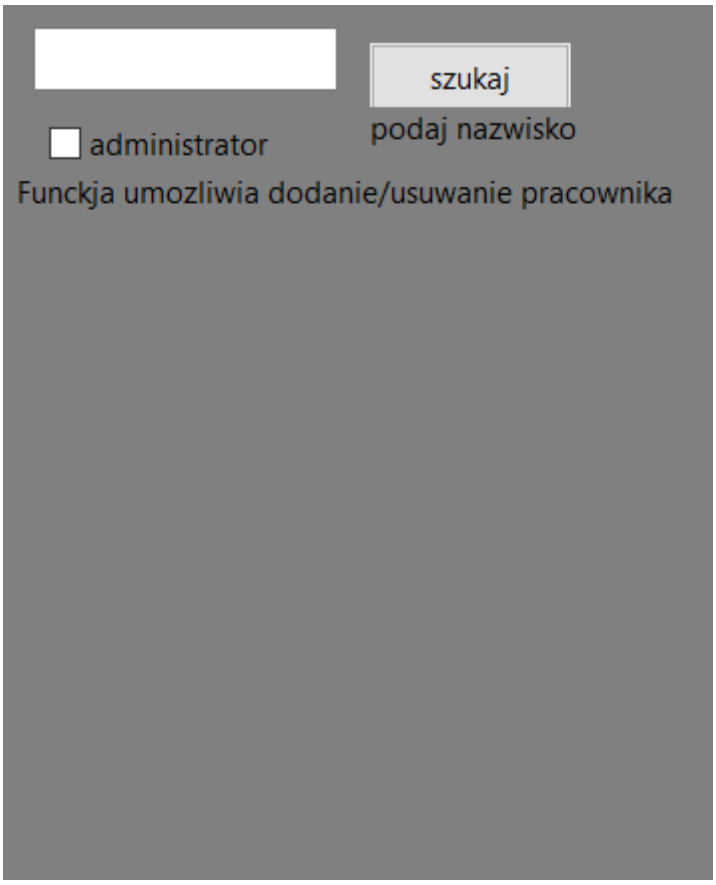
```
41          void OnButton2Click(wxCommandEvent& event);
42          void OnButton1Click(wxCommandEvent& event);
43          //*)
44      public:
45          wxString Funcl(wxString a)
46          {
47              wxString b;
48              if(a=="log")b=log->GetValue();
49              if(a=="pas")b=pas->GetValue();
50              return b;
51          }
52
53          DECLARE_EVENT_TABLE()
54  };
55
56  #endif
57
```

Działanie tego polega na pobraniu zmiennych w ataki sam sposób jak przy opisie wcześniejszego logowania.

Oraz w pliku logowanie.cpp mamy takie dwie funkcję:

```
47
48    void logowanie::OnButton2Click(wxCommandEvent& event)
49    {
50        Destroy();
51
52    }
53
54    void logowanie::OnButton1Click(wxCommandEvent& event)
55    {
56        Destroy();
57    }
58
```

Które po kliknięciu przycisku „Ok” bądź „Cancel” zamkną nasze okno dialogowe potrzebne do zalogowania. Ale w przypadku wxID_OK uruchomi jeszcze kolejne działanie procedur. Wracając do kodu z pra.cpp.



Wszystko pod napisem „Funkcja umozliwia…..” ma ustawioną niewidoczność, podczas poprawnego zalogowania nasze pola stają się widzialne, dzięki funkcji Show(true) dosłownie to tłumacząc znaczy pokaż-prawda.

```
159  void pra::OnButton1Click1(wxCommandEvent& event)
160  {
161
162      wxString a,b,c,t,d,e;
163      a=TextCtrl1->GetValue();
164      b=TextCtrl2->GetValue();
165      c=TextCtrl3->GetValue();
166      d=TextCtrl4->GetValue();
167      e="CREATE TABLE dbpracow(imie TEXT,nazwisko TEXT, telefon INT, specjalizacja TEXT);";
168      t="INSERT INTO dbpracow(imie, nazwisko, telefon, specjalizacja) VALUES ('"+a+"', '"+b+"', "+c+", '"+d+"');";
169      wxString testDBName = wxGetCwd() + wxT("/database.db");
170      db->Open(testDBName);
171      db->ExecuteUpdate(wxString::Format(_("%s"),t));
172      db->Close();
173      StatusBar1->SetStatusText(_("Querry: ")+t);
174      TextCtrl1->SetValue(_(""));
175      TextCtrl2->SetValue(_(""));
176      TextCtrl3->SetValue(_(""));
177      TextCtrl4->SetValue(_(""));
178      OnButton4Click(event);
179  }
```

Ten kod działa analogicznie jak każdy poprzedni dodający wartości do tabeli o nazwie „dbpracow” po kliknięciu przycisku „dodaj pracownika” o nazwie „Button1”.

Czerwony przycisk „refresh” znajdujący się w każdym oknie to nic innego jak ta funkcja:

```
218
219   void pra::OnButton4Click(wxCommandEvent& event)
220   {
221       wxString testDBName = wxGetCwd() + wxT("/database.db");
222       db->Open(testDBName);
223       wxSQLite3ResultSet Res = db->ExecuteQuery(wxString::Format(_("%s"),"SELECT * FROM dbpracow;"));
224       Gridl->ClearGrid();
225       int count = 0;
226
227       while (Res.NextRow())
228       {
229           Gridl->SetCellValue(count,0,wxString::Format(_("%s"),Res.GetAsString(0)) ) ;
230           Gridl->SetCellValue(count,1,wxString::Format(_("%s"),Res.GetAsString(1)) ) ;
231           Gridl->SetCellValue(count,2,wxString::Format(_("%d"),Res.GetInt(2)) ) ;
232           Gridl->SetCellValue(count,3,wxString::Format(_("%s"),Res.GetAsString(3)) ) ;
233           count++;
234       }
235       Res.Finalize();
236       db->Close();
237   }
238
```

2.4 Okienko Klienci

Okno Klienci po zalogowaniu: dodawanie klientów oraz odświeżanie tabeli działa w identyczny sposób jak poprzednio, z małymi drobnymi różnicami w przypadku ilości zmiennych bądź treścią polecenia.

Drobna różnica jest tutaj w przypadku konsoli do zarządzania bazą danych.

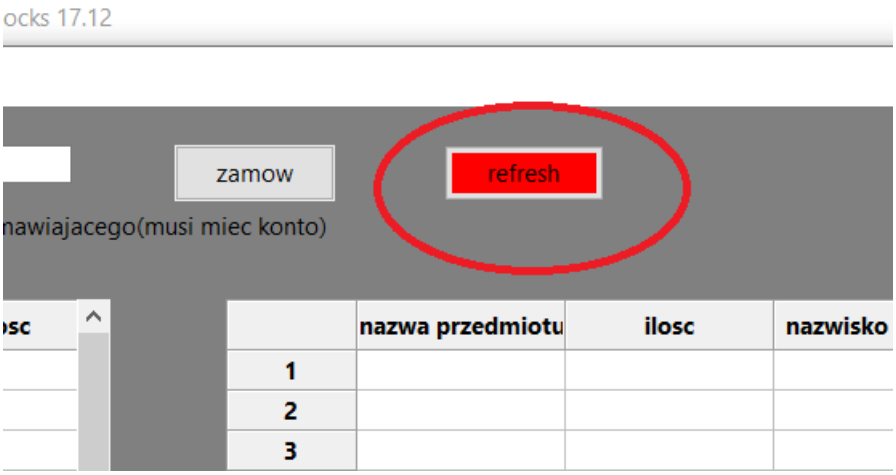
```
void kli::OnButton2Click(wxCommandEvent& event)
{
    wxString a;
    a=TextCtrl5->GetValue();
    wxString testDBName = wxGetCwd() + wxT("/database.db");
    db->Open(testDBName);
    db->ExecuteUpdate(wxString::Format(_("%s"),a));
    db->Close();
    StatusBar1->SetStatusText(_("Querry: ")+a);
    TextCtrl5->SetValue(_(""));
    OnButton3Click(event);
}
```

2.5 Okienko zamówienia.

Tak wygląda to polecenie i różni się tylko tym że z naszego okienka tekstowego pobierane jest całe polecenie, nie ma nic ustalonego na sztywno. Dzięki temu możemy cały zakres poleceń do baz danych.

Pewne komplikacje kodu zaczynają się w naszym ostatnim okienku „zamówienia”

Po kliknięciu tego przycisku „refresh”:



Uruchamia się poniżej zawarty kod programu:

```
88  void zam::OnButton2Click(wxCommandEvent& event)
89  {
90
91      wxString testDBName = wxGetCwd() + wxT("/database.db");
92      db->Open(testDBName);
93      wxSQLite3ResultSet Res = db->ExecuteQuery(wxString::Format(_("%s"),"SELECT * FROM dbpro;"));
94      Grid1->ClearGrid();
95      int count = 0;
96
97      while (Res.NextRow())
98      {
99          Grid1->SetCellValue(count,0,wxString::Format(_("%s"),Res.GetAsString(0)) ) ;
100         Grid1->SetCellValue(count,1,wxString::Format(_("%d"),Res.GetInt(1)) ) ;
101         Grid1->SetCellValue(count,2,wxString::Format(_("%d"),Res.GetInt(2)) ) ;
102         count++;
103
104     }
105
106     wxSQLite3ResultSet Res2 = db->ExecuteQuery(wxString::Format(_("%s"),"SELECT * FROM dbzam;"));
107     Grid2->ClearGrid();
108     count = 0;
109
110     while (Res2.NextRow())
111     {
112         Grid2->SetCellValue(count,0,wxString::Format(_("%s"),Res2.GetAsString(0)) ) ;
113         Grid2->SetCellValue(count,1,wxString::Format(_("%d"),Res2.GetInt(1)) ) ;
114         Grid2->SetCellValue(count,2,wxString::Format(_("%s"),Res2.GetAsString(2)) ) ;
115         Grid2->SetCellValue(count,3,wxString::Format(_("%d"),Res2.GetInt(3)) ) ;
116         count++;
117
118     }
119     Res.Finalize();
120     db->Close();
121 }
```

Działa on analogicznie do poprzednich kodów wyświetlających dane w tabeli, z tym że tutaj musimy utworzyć dwa połączenia z tabelami o innych nazwach „Res” oraz „Res2”. Reszta działa tak samo.

Pod przyciskiem „zamów” kryje się już trochę bardziej skomplikowana funkcja:

```
123  void zam::OnButton1Click(wxCommandEvent& event)
124  {
125
126      wxString a,b,c,t,d,f,produkt,nazwisko, e,suma, minu, dod;
127      int a1, a2, a3, a4;
128     a=TextCtrl1->GetValue();
129     b=TextCtrl2->GetValue();
130     c=TextCtrl3->GetValue();
131      wxString testDBName = wxGetCwd() + wxT("/database.db");
132     db->Open(testDBName);
133     wxSQLite3ResultSet Res = db->ExecuteQuery(wxString::Format(_("%s"),"SELECT * FROM dbpro WHERE nazwa='"+a+"';"));
134     d=Res.GetAsString(1);
135     e=Res.GetAsString(2);
136     produkt=Res.GetAsString(0);
137     if(produkt==a)
138     {
139         wxSQLite3ResultSet Res2 = db->ExecuteQuery(wxString::Format(_("%s"),"SELECT * FROM dbkli WHERE nazwisko='"+c+"';"));
140         f=Res2.GetAsString(3);
141         nazwisko=Res2.GetAsString(1);
142         if(c==nazwisko)
143         {
144             a3=atoi(e.c_str());
145             a2=atoi(b.c_str());
146             if(a2<=a3)
147             {
148                 a1= atoi(d.c_str());
149                 a4=atoi(f.c_str());
150                 dod << (a4+1);
151                 minu << (a3-a2);
152                 suma << (a1*a2);
153                 t="INSERT INTO dbzam(nazwa, ilosc, nazwisko, kwota) VALUES('"+a+"', "+b+", '"+c+"', "+suma+");";
154                 db->ExecuteUpdate(wxString::Format(_("%s"),"UPDATE dbpro SET ilosc ='"+minu+"' WHERE nazwa='"+a+"';"));
```

```
154                 db->ExecuteUpdate(wxString::Format(_("%s"),"UPDATE dbpro SET ilosc ='"+minu+"' WHERE nazwa='"+a+"';"));
155                 db->ExecuteUpdate(wxString::Format(_("%s"),"UPDATE dbkli SET ilosc ='"+dod+"' WHERE nazwisko='"+c+"';"));
156                 db->ExecuteUpdate(wxString::Format(_("%s"),t));
157                 StatusBar1->SetStatusText(_("Querry: ")+t);
158             } else StatusBar1->SetStatusText(_("Za malo produktow w magazynie!! "));
159
160
161         } else StatusBar1->SetStatusText(_("Nie posiadamy takiego uzytkownika w bazie danych!! "));
162
163     } else StatusBar1->SetStatusText(_("Nie ma takiego przedmiotu w naszym magazynie, upewnij sie ze wpisales poprawana nazwe!! "));
164
165
166
167
168
169     db->Close();
170
171     TextCtrl1->SetValue(_(""));
172     TextCtrl2->SetValue(_(""));
173     TextCtrl3->SetValue(_(""));
174     OnButton2Click(event);
175
176 }
```

Tutaj musimy zainicjować już trochę więcej zmiennych w tym również liczby całkowite, ponieważ będzie nam to potrzebne do zmiany wartości z pobranych komórek na liczby typu int. W tym miejscu pojawia się nowa funkcja taka jak w linijce 134 czy 135. Pobiera ona zmienna typu string z wybranej wcześniej tabeli i zapisuje ją do zmiennych. Sa to dane o ocenie i ilości towarów. Pierwszy if sprawdza nam czy podany przez klienta produkt istnieje w naszej bazie danych, jeśli nie wyświetla stosowny komunikat w „StatusBar”. Jeśli tak, przechodzi do dalszej części. Pobiera nam z bazy nazwisko oraz stan jego zamówień klienta. Jeśli nie uda się wyszukaj klienta o danym nazwisku wtedy kolejny if wyświetli nam o tym komunikat. W przeciwnym razie zamieni nam wartość z komórek o ilości produktów w magazynie oraz ilości zamawianych przez klienta na zmienne typu int i sprawdzi czy jest ich wystraczająca ilość do zaakceptowania transakcji. Jeśli Tak to do pobranej ilości zamówień dodamy kolejne, od ilości towaru będącego w magazynie odejmiemy zamówiony oraz, wymnożymy całość należności

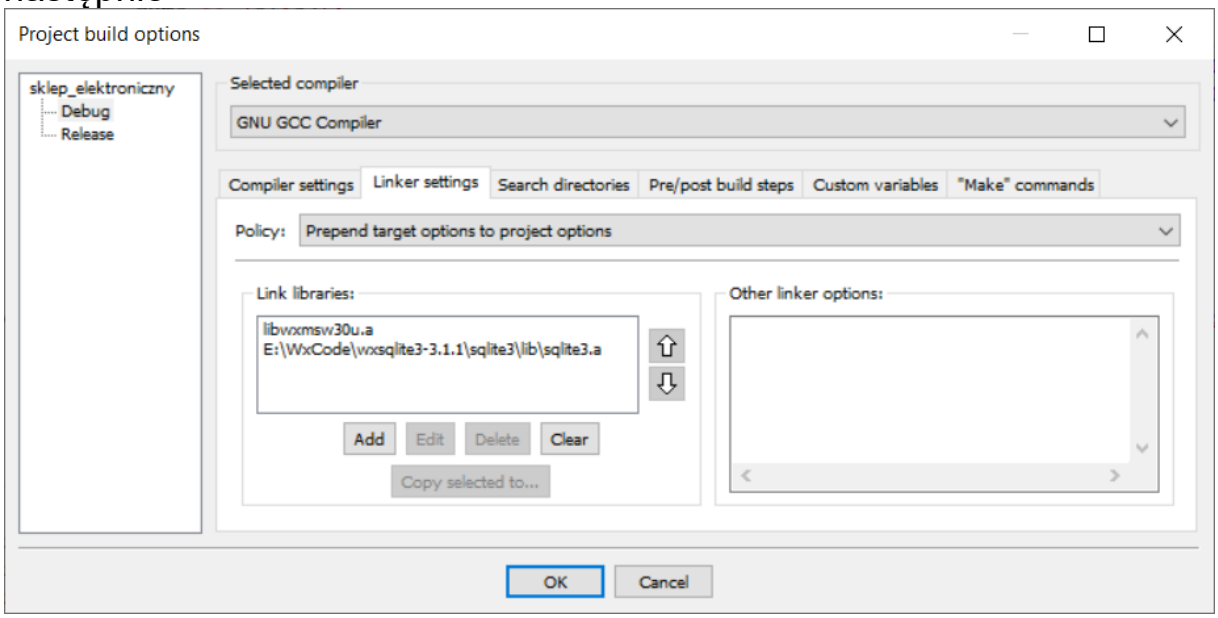
za zamówienie. Wprowadzimy dane do bazy oraz wyświetlimy odpowiedni komunikat w „StatusBar”. Kolejne połączenie z baza jest zamknięte a komórki do wprowadzenia danych czyszczone.

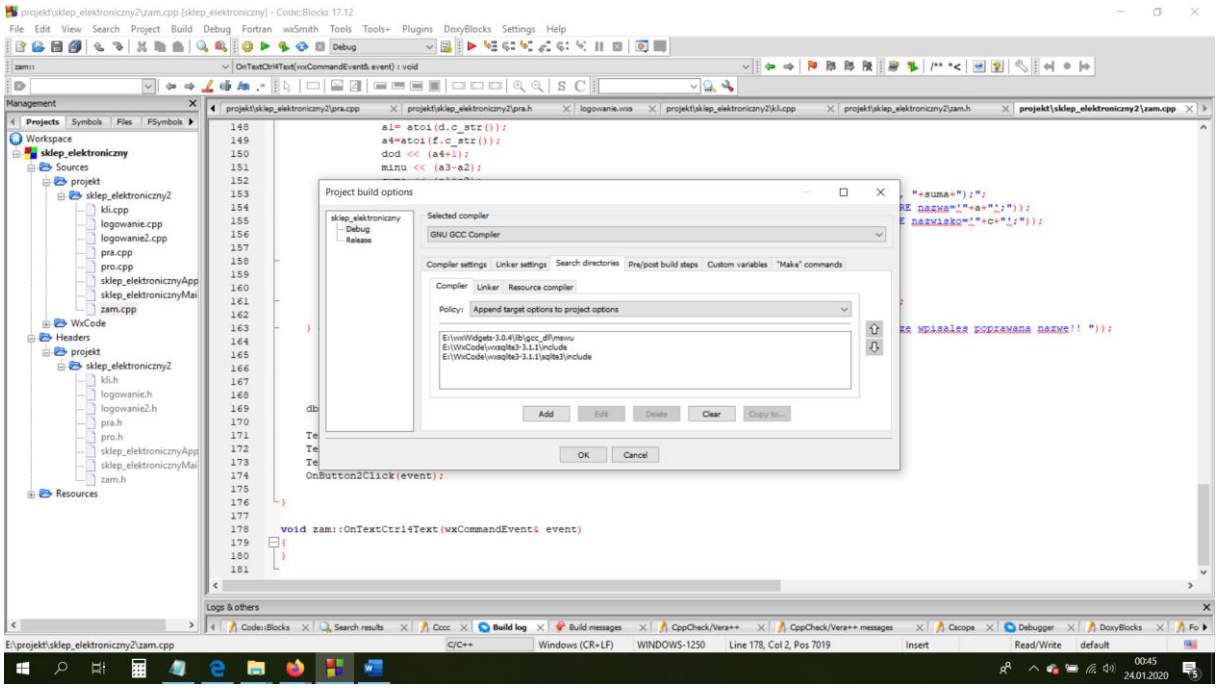
3) Instalacja SQLite3.

Pobieramy plik z tej strony:

https://drive.google.com/file/d/0B1Jtvy4Ey-uwZXRqeVFVa0xLeHM/edit

następnie





Dodajemy takie biblioteki ze ścieżką gdzie umieściliśmy pobrane pliki.

Link do dokumentacji:

https://www.tutorialspoint.com/sqlite/

4) Oświadczenie o samodzielnym wykonaniu ćwiczenia:

Oświadczenie o samodzielnym wykonaniu ćwiczenia:

Oświadczam, że sprawozdanie z ćwiczenia pt.: „*tytuł ćwiczenia*" zostało wykonane przeze mnie/przez nasz zespół osobiście. Zamieszczone w sprawozdaniu wyniki badań zostały uzyskane przeze mnie/przez nasz zespół podczas wykonywania ćwiczenia laboratoryjnego.

…….

Podpis osoby/osób wykonujących sprawozdanie