



WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Programowanie

Ignacy Tokarz

Nr albumu studenta w67657

***System wspomagający obsługę klienta oraz
zarządzanie magazynem w kafejce internetowej***

Praca projektowa Szkolenie Techniczne 1

Rzeszów 2024

Spis treści

Wstęp	4
Rozdział 1.....	5
Opis założeń projektu	5
1.1 Cele projektu	5
1.2 Wymagania funkcjonalne i нефункционалне	5
1.3 Wymagania sprzętowe i programowe	6
Rozdział 2.....	7
Warstwa użytkowa projektu	7
2.1 Ogólny wygląd aplikacji	7
2.2 Zakładka Sprzedaż	7
2.3 Zakładka Magazyn	10
2.4 Zakładka Stan kasy	11
2.5 Zakładka Koniec dnia	12
2.6 Kod źródłowy dla Programu	12
Rozdział 3.....	13
Klasy wspomagające	13
3.1 Klasa Functions, komunikacja z bazą danych.....	13
3.2 Klasa Validator, walidacja danych	14
3.3 Klasa Koszyk, odświeżanie formularza	16
Rozdział 4.....	17
Podsumowanie.....	17
Bibliografia.....	18

Wstęp

W dzisiejszym dynamicznym środowisku biznesowym, gdzie czas jest cennym zasobem, skomplikowane systemy zarządzania kasą mogą stwarzać znaczne utrudnienia w codziennej pracy. Zrozumienie potrzeb przedsiębiorców oraz pracowników branży usługowej, zwłaszcza tych związanych z ofertą przekąsek, napojów i usług w nowoczesnych kafejkach internetowych, stało się kluczowym punktem wyjścia do stworzenia efektywnego narzędzia. Wychodząc naprzeciw tym potrzebom, powstał program, który nie tylko eliminuje zawichości związane z obsługą kasy, lecz także dostarcza prostą i intuicyjną aplikację dla sprzedawców.

Aplikacja została starannie zaprojektowana z myślą o instytucjach oferujących szeroką gamę produktów oraz usług, szczególnie skupiając się na nowoczesnych kafejkach internetowych. Zdecydowanie aplikacja skupia się na dostarczeniu narzędzia, które nie tylko ułatwia proces sprzedaży, ale także efektywnie wspomaga zarządzanie magazynem i obsługą kasy fiskalnej. Wierze, że nasz program nie tylko usprawni codzienną pracę, ale także przyczyni się do wzrostu efektywności działania przedsiębiorstw z branży.

Aplikacja zawiera kompleksowy zestaw funkcji, które umożliwiają nie tylko sprawną realizację transakcji, ale również skuteczne monitorowanie stanu magazynowego. Aplikacja nie jest jedynie narzędziem do obsługi kasy; to kompleksowe rozwiązanie, które integruje funkcje sprzedaży, zarządzania magazynem i generowania raportów. W tym dokumencie szczegółowo omówię kluczowe aspekty mojego programu, prezentując korzyści, jakie niesie dla nowoczesnych kafejek internetowych oraz innych podobnych instytucji. Aplikacja dąży do nie tylko funkcjonalności, ale również łatwej obsługi.

Rozdział 1

Opis założeń projektu

1.1 Cele projektu

Głównym celem programu jest prosta i łatwa w obsłudze aplikacja, która dodatkowo będzie posiadać wszystkie najważniejsze i najpotrzebniejsze funkcje tak, aby pracownicy bez problemowo mogli zająć się klientem. Z doświadczenia wiemy jak źle zrobiony program potrafi utrudnić codzienne obowiązki w firmie, przy pracy z klientem program musi być szybki, oraz niezawodny, aby to osiągnąć moja aplikacja została stworzona w sposób minimalistyczny. Nie zawiera żadnych zbędnych funkcji. Postawiłem na prosty interfejs graficzny z krótkim oraz prostym menu, które zawiera wszystkie najpotrzebniejsze funkcje niezbędne do pracy.

Program zawiera 4 główne zakładki:

- Sprzedaż
- Magazyn
- Stan Kasy
- Koniec Dnia

Każda z nich zawiera dostęp do innych funkcjonalności projektu

1.2 Wymagania funkcjonalne i нефункционалне

Wymagania funkcjonalne

- Program posiada 1 bazę danych Sql Server oraz w niej 4 tablice
- Koniec dnia
 - Każda transakcja, która przechodzi przez system jest zapisywana w tablicy "Utarg" gdzie pokazane są szczegóły każdej wykonanej transakcji do momentu zakończenia dnia. Po wybraniu tej opcji użytkownik posiada wgląd w transakcje oraz przycisk do zakończenia dnia.
- Stan Kasy
 - Kasa zapisuje ilość pobranej gotówki lub kwoty transakcji kartą płatniczą, oraz posiada pole gdzie sumuje obie metody płatności. Dzięki temu wiemy ile mamy pieniędzy w systemie oraz mamy wgląd również do poszczególnych sprzedanych przedmiotów oraz usług w danym dniu.
- Magazyn
 - Magazyn posiada 5 zasadniczych funkcjonalności, wyświetlanie stanów, dodawanie stanu do istniejącego produktu, dodawanie nowego, usunięcie produktu z magazynu oraz edycja pozycji w magazynie.
 - **Dodawanie stanu do istniejącego produktu** – Użytkownik wybiera z listy produkt do którego chce dodać stan następnie wpisuje ilość jaką chce dodać.
 - **Usuń produkt** – Użytkownik wybiera z listy produkt do usunięcia z bazy danych
 - **Dodaj nowy produkt do magazynu** – Użytkownik wpisuje nazwę nowego produktu jaki chce dodać, następnie wybiera z listy kategorie pozycji, deklarowaną ilość oraz cenę netto produktu, z którego później będzie wyliczana cena brutto podczas sprzedaży produktów

- **Edytuj produkt** – Użytkownik wybiera z listy produkt do edycji następnie może edytować każde pole przedmiotu z wyłączeniem nazwy.
- Sprzedaż
 - Okno sprzedaży posiada 3 główne kategorie na sprzedaż, po wybraniu jednej z nich nastąpi wyświetlenie produktów ich cen oraz stanów magazynowych, żeby wybrać pozycję, którą chcemy dodać do koszyka użytkownik musi wybrać z listy produkt oraz wybrać ilość jaką chce dodać do koszyka
 - **Koszyk**
 - Przedmioty dodawane do koszyka będą wyświetlać się w postaci listy z nazwą, ilością oraz ceną za jedną sztukę, suma do zapłaty na cały koszyk jest wyświetlana niżej w polu „suma”.
 - **Zapłać** – Po skompletowaniu zamówienia przycisk ten otworzy wybór płatności
 - **Anuluj zakupy** – Czyści koszyk oraz zwraca produkty na magazyn
 - **Zmień ilość produktu** – Użytkownik ma możliwość poprawienia błędnie wprowadzonych pozycji do koszyka, należy wybrać z listy przedmiot do edycji oraz wprowadzić nową ilość produktu
 - **Usuń produkt z koszyka** - Użytkownik ma możliwość usunięcia produktu z koszyka po wybraniu go z listy i wciśnięciu przycisku.
- Wymagania sprzętowe mojego programu są minimalne, każdy komputer z aktualnie dostępnym procesorem poradzi sobie z obsługą programu

Wymagania niefunkcjonalne

- Program posiada prosty i przyjazny dla użytkownika interfejs graficzny z opisanymi wszystkimi funkcjami.
- Aktualna wersja programu jest przygotowana pod potencjalną rozbudowę o kolejne produkty, kolejne rodzaje usług, bez problemu również można do niego dodać kolejne funkcjonalności.
- Bezpieczeństwo aplikacji jest na wysokim poziomie, ponieważ wszystko odbywa się ze strony klienta i nie jest w żaden sposób połączone z Internetem czy innymi komputerami, jedyną rzeczą jaką pracownicy muszą zadbać to przypilnować pliki baz danych.
- Program jest również zabezpieczony na wypadek nieoczekiwanego zamknięcia komputera lub programu, wszystkie zmiany na bieżąco się zapisują po nagłym wyłączeniu dane nie zostaną utracone.

1.3 Wymagania sprzętowe i programowe

Minimalne wymagania systemowe do poprawnego działania programu

- 60 MB wolnego miejsca na dysku
- Procesor o częstotliwości taktowania 1Ghz lub więcej
- 1 GB pamięci RAM

Obsługiwane systemy operacyjne

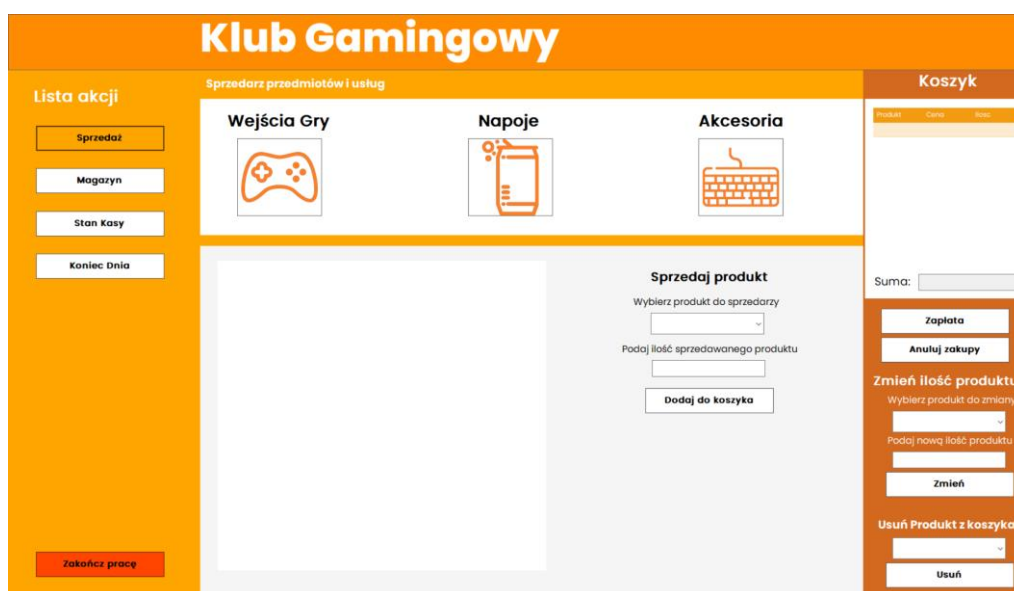
- Windows 10 | System 64-bit
- Windows 11 | System 64-bit

Rozdział 2

Warstwa użytkowa projektu

2.1 Ogólny wygląd aplikacji

Aplikacja zawiera kolorystykę nawiązującą do kolorystyki firmy, dla której program został stworzony. Po lewej stronie znajduje się główne menu, które będzie towarzyszyć użytkownikowi przez całą pracę z aplikacją, w dowolnej chwili możemy przetaczać pomiędzy kluczowymi stronami tj. „Sprzedaż”, „Magazyn”, „Stan kasy”, „Koniec Dnia” oraz przycisk do wyłączenia aplikacji. W środkowej części interfejsu będą się pojawiać narzędzia do wykonywania konkretnych czynności powiązanych z operacją główną jaką wybierzemy na poniższym obrazku widnieje przykładowe okno „Sprzedaż”.



Rysunek 1 Wygląd ogólny aplikacji

2.2 Zakładka Sprzedaż

Wybór rodzaju usługi

Podczas sprzedaży użytkownik musi wybrać jaki rodzaj „przedmiotu” chce sprzedać. Po wybraniu wyświetli się lista produktów.



Rysunek 2 Wybór rodzaju usługi

Wybór przedmiotu

Lista zawiera informacje o produktach na magazynie, ich ilości oraz cenie brutto, która według założeń firmy jest wyliczana z ceny netto plus 60% marży na produkcie i zaokrąglone do pełnej

kwoty (int). Następnie po prawej stronie użytkownik wybiera z listy, który produkt chce dodać do koszyka oraz jaką ilość tego przedmiotu, jeżeli wybierze przedmiot, którego nie ma lub wybierze za dużą ilość program poinformuje o błędzie związanym z problemem i poprosi o poprawienie wartości. Kiedy użytkownik zatwierdzi dane klikając w przycisk zostaną one zabrane z magazynu oraz przeniesione do tymczasowej tablicy „Koszyk”.

Usługa	Stan_magazynu	Cena
cola	60	6
Monster	0	7
Sprite	12	5
Burn	12	5
Woda	19	5
Sok Jablkowy	30	6

Sprzedaj produkt

Wybierz produkt do sprzedazy

Podaj ilość sprzedawanego produktu

Dodaj do koszyka

Rysunek 3 Wybór przedmiotu do sprzedaży

Obsługa koszyka

Produkty dodane do koszyka znajdują się w tablicy z podanymi ilościami oraz ceną brutto za jedną sztukę, na dole tabeli znajdują się suma koszyka jaką klient musi zapłacić. Koszyk posiada dwie specjalne funkcjonalności zmiana ilości produktu oraz usuwanie produktu oraz dwa główne przyciski „Anuluj zakupy”, który czyści koszyk oraz zwraca wzięte przedmioty do magazynu i przycisk „Zapłata”, która kieruje do następnego okna z wyborem zapłaty.

Koszyk

Produkt	Cena	Ilość
cola	6	2
Burn	5	3

Suma: 27

Zapłata

Anuluj zakupy

Zmień ilość produktu

Wybierz produkt do zmiany

Podaj nową ilość produktu

Zmień

Usuń Produkt z koszyka

Usuń

Rysunek 4 Koszyk oraz opcje edycji

Wybór sposobu płatności

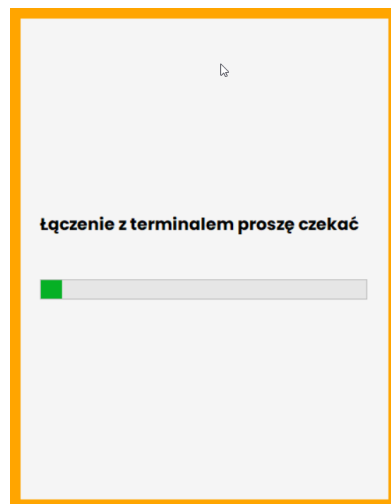
Pojawia się okno z wyborem sposobu opcjonalnie do anulowania, po wyborze sposobu płatności, „nie ma odwrotu”.



Rysunek 5 Wybór sposobu płatności

Płatność kartą

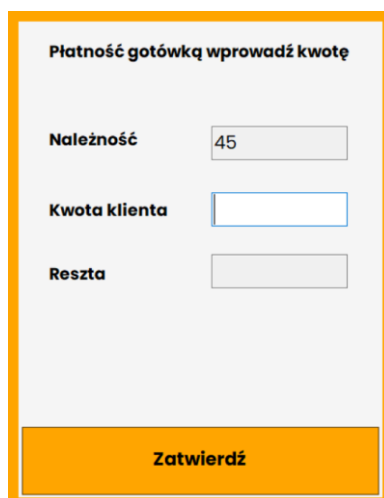
Program łączy się z terminalem płatniczym, jeżeli transakcja zostanie zaakceptowana okno się zamknie oraz użytkownik zostanie przeniesiony na widok „Sprzedaż”.



Rysunek 6 Płatność kartą

Płatność gotówką

Program wyświetla kalkulator do wydania reszty, w czasie rzeczywistym przeliczą resztę, jeżeli reszta będzie na minusie prze pozwoli przejść dalej.



Płatność gotówką wprowadź kwotę

Należność 45

Kwota klienta

Reszta

Zatwierdź

Rysunek 7 Płatność gotówką

2.3 Zakładka Magazyn

Wygląd magazynu

Magazyn został podzielony na 5 sektorów.

Dodawanie stanu do istniejącego produktu

Użytkownik wybiera z listy produktów, produkt, do którego chce stan i następnie zatwierdza przyciskiem „Dodaj stan”.

Usuwanie produktu

Użytkownik wybiera produkt z listy, który chce usunąć, następnie zatwierdza przyciskiem „Usuń”.

Dodawanie nowego produktu do magazynu

Użytkownik musi wypełnić wszystkie pola zgodnie z podpisami, oraz wybrać jedną z trzech kategorii, do późniejszej klasyfikacji w zakładce „Sprzedaż

Edytowanie produktu

Użytkownik ma możliwość edycji każdego produktu oraz wszystkich jego parametrów z wyłączeniem nazwy. Po wybraniu produktu z listy wszystkie pola automatycznie się uzupełnią aktualnymi danymi, a użytkownik zmienia tylko wartości, które chce zmienić a następnie zatwierdza zmiany przyciskiem „Uaktualnij produkt”

Wgląd w tablicę magazynu

Po prawej stronie wyświetlają się aktualne wartości z magazynu, które są aktualizowane w czasie rzeczywistym

Zarządzanie magazynem

Dodaj stan do istniejącego produktu

Nazwa produktu

Dodawana ilość

Dodaj stan

Dodaj nowy produkt do magazynu

Nazwa nowego produktu

Kategoria

Deklarowana ilość

Cena Netto produktu

Dodaj Produkt

Usuń produkt

Nazwa produktu

Usuń

Edytuj Produkt

Wybierz produkt do edycji

Nowa Kategoria

Nowa ilość magazynowa

Nowa Cena netto

Uaktualnij produkt

Aktualne stany magazynowe

Nazwa	Ilość	Cena_Netto	Kategoria
cola	56	4,23	Napoje
Monster	0	4,5	Napoje
pcgaming	0	19	Usługa
Klawiatura	0	89,4	Akcesoria
Sprite	12	3,66	Napoje
Burn	6	3,7	Napoje
Woda	19	3,2	Napoje
Sok Jablkowy	27	4,2	Napoje

Rysunek 8 Zakładka Magazyn

2.4 Zakładka Stan kasy

Utarg – Dzisiaj

Tabela ta pokazuje wszystkie transakcje, które się dzisiaj odbyły sumę sprzedanych przedmiotów oraz ilość zarobionych pieniędzy z danego przedmiotu. Tabela ta jest potrzebna do wydrukowania dokumentu podczas zamknięcia dnia. Po zakończeniu dnia tabela ta zostaje wyczyszczona.

Ogólna ilość zarobionych pieniędzy

Tabela pokazuje całkowitą zarobioną sumę pieniędzy od rozpoczęcia działalności. W niej możemy zobaczyć ile pieniędzy firma zarobiła przy płatnościami kartą czy gotówką oraz łączną sumę zarobionych pieniędzy.

Stan kasy – sprawdź obecny stan kasy oraz utarg				
Utarg – Dzisiaj			Ogólna ilość zarobionych pieniędzy	
Przedmiot	Ilość	Suma_zarobiona	Rodzaj	Suma
cola	4	24	Karta	8934
Burn	6	30	Gotówka	447
Sok Jablkowy	3	18	Suma ogólna	9381

Rysunek 9 Zakładka Stan kasy

2.5 Zakładka Koniec dnia

Zakładka pozwala na ponowne wyświetlenie tabeli „Utarg - Dzisiaj” oraz na rozpoczęciu procedury kończenia dnia. Użytkownik może zakończyć dzień klikając w przycisk „Drukuj raport” program zajmie się resztą i przygotuje dokument dla kasy fiskalnej na podstawie tabeli, która wyświetla się po prawej.

Aby zakończyć dzień wciśnij przycisk oraz postępuj zgodnie z instrukcjami		
Drukuj raport		
Całkowity utarg dzisiejszy dzień		
przedmiot	ilosc	suma_zarobiona
cola	4	24
luzn	6	30
Sok Jablkowy	3	18

Rysunek 10 Zakładka Koniec dnia

2.6 Kod źródłowy dla Programu

Kod źródłowy ze szczegółami znajduje się na platformie github na koncie użytkownika B0b0lin lub pod linkiem : https://github.com/B0b0lin/Projekt_sklep_v2

Pod linkiem znajduje się ta dokumentacja oraz plik bazy danych typu Sql Server, który w podstawowej wersji programu działa na zasadzie localhost.

Rozdział 3

Klasy wspomagające

3.1 Klasa Functions, komunikacja z bazą danych

Klasa wspomagająca Functions odpowiada za łączenie się z bazą danych, wysyłanie oraz pobieranie danych.

GetData i SetData

Te metody jak nazwa wskazuje, wysyłają podstawowe zapytanie lub edycje do bazy danych

```
internal class Functions
{
    private SqlConnection Con;
    private SqlCommand Cmd;
    private DataTable dt;
    private SqlDataAdapter sda;
    private string ConStr;
    // 8 references
    public Functions()
    {
        //properties na bazie danych >> "connection string" wkleic tutaj na innym uzadzeniu ze dzialka
        ConStr = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\ignac\Documents\Magazyn.mdf;Integrated Security=True;Connect Timeout=30";
        Con = new SqlConnection(ConStr);
        Cmd = new SqlCommand();
        Cmd.Connection = Con;
    }

    // 7 references
    public DataTable GetData(string Query)
    {
        dt = new DataTable();
        sda = new SqlDataAdapter(Query, ConStr);
        sda.Fill(dt);
        return dt;
    }

    // 22 references
    public int SetData(string Query)
    {
        int cnt = 0;
        if(Con.State == ConnectionState.Closed)
        {
            Con.Open();
        }
        Cmd.CommandText = Query;
        cnt = Cmd.ExecuteNonQuery();
        Con.Close();
        return cnt;
    }
}
```

Rysunek 11 Klasa Functions

GetStringData

Metoda, która zwraca wynik zapytania w formie stringa

```
17 references
public string GetStringData(string Query)
{
    if (Con.State == ConnectionState.Closed)
    {
        Con.Open();
    }
    SqlCommand cmd = new SqlCommand(Query, Con);
    SqlDataReader DR = cmd.ExecuteReader();
    while (DR.Read())
    {
        string temp = Convert.ToString(DR[0]);
        Con.Close();
        return temp;
    }
    Con.Close();
    return "";
}
```

Rysunek 12 Klasa Functions

GetDataToList

Metoda, która zwraca wynik zapytania w formie listy stringów, np. nazw

```
4 references
public List<string> GetDataToList(string Query)
{
    if (Con.State == ConnectionState.Closed)
    {
        Con.Open();
    }
    SqlCommand cmdd = new SqlCommand(Query, Con);
    SqlDataReader DR = cmdd.ExecuteReader();

    List<string> list = new List<string>();
    do
    {
        int count = DR.FieldCount;
        while (DR.Read())
        {
            for (int i = 0; i < count; i++)
            {
                list.Add(Convert.ToString(DR.GetValue(i)));
            }
        } while (DR.NextResult());
    } while (DR.NextResult());
    Con.Close();
    return list;
}
```

Rysunek 13 Klasa Functions

3.2 Klasa Validator, walidacja danych

Klasa Validator sprawdza wprowadzane dane oraz wykonuje pewne czynności związane z dodawaniem lub usuwaniem stanów w magazynie

MagazynCheckSub

Metoda sprawdza czy wartości, które chce podać użytkownik są odpowiednie oraz po weryfikacji wykonuje update tablicy magazyn

```
3 references
internal class Validator
{
    Functions Con;
    1 reference
    public Validator()
    {
        Con = new Functions();
    }
    //TODO: Validacja sprzedarzy, kiedy klikamy zaplata musi nastapic walidacja
    //czy pożądana ilość produktu jest w magazynie jak nie to komunikat i rollback
    2 references
    public bool MagazynCheckSub(string item, int quantity)
    {
        string Querymagazyn = $"select quantity from magazyn where name = '{item}'";
        string QueryCategory = $"select category from magazyn where name = '{item}'";

        var quantityMagazyn = Convert.ToInt16(Con.GetStringData(Querymagazyn));

        if (Con.GetStringData(QueryCategory) != "Usługa")
        {
            if (quantity <= quantityMagazyn)
            {
                string Query = $"update magazyn set quantity = quantity - {quantity} where name = '{item}'";
                Query = string.Format(Query);
                Con.SetData(Query);
                return true;
            }
            else
            {
                return false;
            }
        }
        else
        {
            return true;
        }
    }
}
```

Rysunek 14 Klasa Validator

MagazynCheckAdd

Metoda ta działa tak samo jak wcześniejsza tylko dodaje wartości do magazynu ta metoda posiada jedno przeciążenie, które było potrzebne do prawidłowego działania programu przy przenoszeniu produktów.

```
2 references
public bool MagazynCheckAdd(string item, int quantity)
{
    string QueryCategory = $"select category from magazyn where name = '{item}'";
    string QueryQuantity = $"Select quantity from koszyk where nazwa_prod = '{item}'";
    string number = Con.GetStringData(QueryQuantity);

    if (Con.GetStringData(QueryCategory) != "Usługa")
    {
        string Query = $"update magazyn set quantity = quantity + {number} where name = '{item}'";
        Query = string.Format(Query);
        Con.SetData(Query);

        string Query1 = $"update magazyn set quantity = quantity - {quantity} where name = '{item}'";
        Query1 = string.Format(Query1);
        Con.SetData(Query1);

        return true;
    }
    else
    {
        return true;
    }
}
```

Rysunek 15 Klasa Validator

```
1 reference
public bool MagazynCheckAdd(string item)
{
    string QueryCategory = $"select category from magazyn where name = '{item}'";
    string QueryQuantity = $"Select quantity from koszyk where nazwa_prod = '{item}'";
    string number = Con.GetStringData(QueryQuantity);

    if (Con.GetStringData(QueryCategory) != "Usługa")
    {
        string Query = $"update magazyn set quantity = quantity + {number} where name = '{item}'";
        Query = string.Format(Query);
        Con.SetData(Query);

        return true;
    }
    else
    {
        return true;
    }
}
```

Rysunek 16 Klasa Validator

3.3 Klasa Koszyk, odświeżanie formularza

Nazwa klasy możliwa, że nie jest odpowiednia natomiast metody już tak, odświeżają one formularz po wykonaniu czynności na tablicy, aby ta odświeżała się w czasie rzeczywistym.

```
3 references
public class Koszyk
{
    2 references
    public static void Refresh()
    {
        for (int i = Application.OpenForms.Count - 1; i >= 0; i--)
        {
            if (Application.OpenForms[i].Name == "Sprzedaz")
                Application.OpenForms[i].Close();
        }

        Sprzedaz obj = new Sprzedaz();
        obj.Show();
    }
}

1 reference
public static void RefreshKoniecDnia()
{
    for (int i = Application.OpenForms.Count - 1; i >= 0; i--)
    {
        if (Application.OpenForms[i].Name == "KoniecDnia")
            Application.OpenForms[i].Close();
    }

    KoniecDnia obj = new KoniecDnia();
    obj.Show();
}
}
```

Rysunek 17 Klasa Koszyk

Rozdział 4

Podsumowanie

Prezentuje najnowszą platformę do obsługi kasy i zarządzania magazynem, która obecnie znajduje się w fazie 1.0. Jednakże, to jedynie pierwszy krok w fascynującej podróży, ponieważ projekt będzie stale rozwijany, kładąc nacisk na poszerzanie funkcjonalności oraz podnoszenie jakości oprawy graficznej.

Jednak plany na rozwój nie ograniczają się jedynie do technicznych ulepszeń. W dążeniu do stworzenia kompleksowego i zintegrowanego narzędzia, w planach jest również dodanie funkcji raportowania. Użytkownicy muszą mieć dostęp do szczegółowych raportów dotyczących sprzedaży, stanu magazynowego czy preferencji klientów, co pomoże im podejmować bardziej świadome decyzje biznesowe.

Finalnie, aplikacja będzie nie tylko funkcjonalna, ale także estetyczna dzięki oprawie graficznej. Interfejs użytkownika jest projektowany z myślą o prostocie obsługi, eliminując zbędne elementy i skupiając się na ergonomicznym designie. Celem jest stworzenie oprogramowania nie tylko funkcjonalnego, ale i przyjemnego w codziennym użytkowaniu.

Podsumowując, kafejki internetowe nieustannie rozwija się i ewoluują, aby sprostać oczekiwaniom dynamicznego rynku, dlatego w duchu motto aplikacji, program zostanie zaprojektowany tak, aby był łatwy, niezawodny oraz przyjemny w użytkowaniu.

Kod źródłowy znajduje się pod linkiem w repozytorium GitHub:

https://github.com/B0b0lin/Projekt_sklep_v2

Bibliografia

- <https://learn.microsoft.com/pl-pl/dotnet/api/system.data.sqlclient.sqldatareader?view=netframework-4.8.1>
- <https://learn.microsoft.com/pl-pl/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022>
- Jacek Matulewski, *C\#: lekcje programowania: praktyczna nauka programowania dla platform .NET i .NET Core*, Helion, Gliwice 2021
- Joseph Albahari, Eric Johanssen, *C\# 8.0 w pigułce*, Helion, Gliwice, 2021
- R. S. Miles, *C\#: zacznij programować!*, Helion, Gliwice, 2020