

<b>BD</b>	<b>Dokumentacja projektu</b>
<b>Autorzy</b>	Mateusz Bocak 125104 Jakub Jakubowski 125125 Kacper Bułaś 127754
<b>Kierunek, rok</b>	Informatyka, II rok, st. Stacjonarne (3,5-l)
<b>Temat Projektu</b>	Kasa samoobsługowa – SSC (Self Service Checkout)

## Spis treści

Wstęp.....	3
Uruchomienie .....	3
Zastosowane technologie.....	4
Opis działań.....	5
Kacper.....	5
Jakub.....	5
Mateusz .....	6
Podział obowiązków w skrócie .....	6
Baza danych .....	7
Funkcjonalności .....	9
Użytkownik .....	9
Kasjer .....	17
Administrator .....	20
Wykorzystanie języka proceduralnego .....	24
Podsumowanie .....	30

## Wstęp

Projekt aplikacji „SSC” jest przeznaczony do użytku na wyspecjalizowanych komputerach do ciągłej pracy jako interaktywna kasa fiskalna. Znacząco ułatwia i przyspiesza obsługę klientów w sklepach zagranicznych.

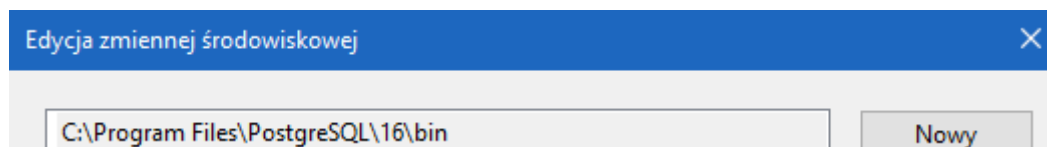
Aplikacja umożliwia dodawanie przedmiotów z wybranych kategorii, dodawanie produktów za pomocą kodu kreskowego, finalizację zakupów, obsługę produktów dostępnych dla osób powyżej 18 roku życia.

Do dyspozycji użytkownika jest również panel kasjera, do którego dostęp mają pracownicy sklepu. Mogą zalogować się do niego poprzez guzik w lewym górnym rogu ekranu. Z panelu pracownika, kasjer może pomóc usunąć produkt z wirtualnego koszyka, zatwierdzić wiek kupującego oraz edytować przedmioty w znajdujące się w koszyku.

Dodatkowo utworzony został panel administratora dostępny z ekranu startowego aplikacji dla użytkowników o specjalnej roli. Z niego administrator może przeprowadzać operacje dodawania, edycji, usuwania oraz wyświetlania produktów, użytkowników oraz pracowników z bazy danych.

## Uruchomienie

Do uruchomienia aplikacji potrzebna jest działający system zarządzania bazami danych – PostgreSQL, który znajduje się w zmiennych środowiskowych dostępnego dla obecnego użytkownika na systemie operacyjnym serii Window.



*Rysunek 1 - dodanie PostgreSQL do zmiennych środowiskowej PATH*

Aby uruchomić aplikację należy skorzystać z dostarczonego skryptu – setup.bat. Uruchomi on tworzenie bazy danych o nazwie SSCDB dla podanego użytkownika bazy danych, zapyta użytkownika o dane logowania do systemu bazy danych oraz utworzy bazę danych z przykładowymi wartościami. Dodatkowo uruchomi instalację aplikacji.

Po zainstalowaniu aplikacji należy uruchomić ją przez utworzony na pulpicie skrót.

## Zastosowane technologie

- [Git](#) – System kontroli wersji używany do śledzenia zmian w kodzie aplikacji i współpracy programistów.
- [Github](#) – Platforma do hostowania kodu źródłowego projektu i współpracy nad nim.
- [Visual Studio](#) – IDE stworzone przez Microsoft do pracy nad projektami z technologii .NET.
- [C#](#) (.NET 8.0) - Wieloparadymatowy, obiektowy język programowania z rodziny języków C-podobnych stworzony przez Microsoft.
- [WindowsForms](#) – framework do tworzenia GUI w technologiach .NET.
- [Npgsql](#) – ORM (Object Relational Mapping) framework dla technologii .NET, konkretnie dla baz danych PostgreSQL. Pozwala na programatyczny dostęp do bazy danych w kodzie.
- [PostgreSQL](#) – Otwarty system do zarządzania relacyjnymi bazami danych.

## Opis działań

Na samym początku projektu ustaliliśmy podstawowe cele, które powinien spełniać projekt. Podzieliliśmy je na mniejsze zadania, a następnie między siebie, korzystając z GitHub Projects.

Staraliśmy podzielić obowiązki między siebie i pracować w trybie GitHub flow, tworząc oddzielne branche na dzień pracy.

Po ukończeniu prac mergowaliśmy gotowe zmiany do gałęzi głównej za pomocą pull requestów na platformie GitHub. W międzyczasie przynajmniej dwóch członków zespołu dyskutowało na temat wprowadzonych zmian w celu lepszej orientacji w projekcie.

Prace zostały podzielone w następujący sposób: Jakub – UI, Mateusz i Kacper – logika biznesowa, CRUD, walidacje.

### Kacper

Kacper skupił się głównie na backendzie oraz logice aplikacji. Jego obowiązki obejmowały:

- Implementację i optymalizację logiki biznesowej aplikacji.
- Prace nad komunikacją z bazą danych, w tym tworzenie, kontekstu bazy danych oraz komunikację z Npgsql Entity Framework.
- Obsługę autoryzacji użytkowników.
- Obsługa płatności.
- Obsługa dodawania nowych użytkowników.
- Zaprojektowanie funkcji, procedur, wyzwalaczy w języku PL/pgSQL.
- Testowanie funkcjonalności systemu jako całości oraz przeprowadzanie testów akceptacyjnych.
- Implementacja komunikacji z funkcjami proceduralnymi.

### Jakub

Jakub skoncentrował się na interfejsie użytkownika oraz implementacji funkcji w Windows Forms. Jego obowiązki obejmowały:

- Zaprojektowanie relacyjnej bazy danych
- Projektowanie i implementację interfejsu użytkownika przy użyciu Windows Forms.
- Obsługa kontrolek w Windows Forms oraz ich integracji z logiką aplikacji.
- Obsługa walidacji dla interfejsu użytkownika
- Tworzenie i stylizację formularzy oraz kontrolek interfejsu.
- Integrację interfejsu użytkownika z logiką aplikacji.
- Implementacja operacji CRUD.
- Testowanie i poprawianie błędów w warstwie interfejsu użytkownika, aby zapewnić spójne i intuicyjne doświadczenie użytkownika.
- Testowanie funkcjonalności systemu jako całości oraz przeprowadzanie testów akceptacyjnych.
- Zaprojektowanie wyzwalaczy, kursorów języka proceduralnego dla bazy danych
- Implementacja obsługi języka proceduralnego w aplikacji po stronie użytkowej

## Mateusz

Mateusz zajmował się zarówno UI, jak i backendem, pełniąc rolę wsparcia dla obu obszarów. Jego obowiązki obejmowały:

- Udział w implementacji logiki aplikacji oraz integracji z bazą danych.
- Integrację interfejsu użytkownika z logiką aplikacji.
- Implementacja operacji CRUD.
- Zaprojektowanie funkcji, procedur, wyzwalaczy proceduralnych w języku PL/pgSQL.
- Obsługa kontrolek w Windows Forms oraz ich integracji z logiką aplikacji.
- Konfiguracja frameworku ORM do komunikacji z bazą.
- Debugowanie i rozwiązywanie problemów na styku logiki aplikacji i interfejsu użytkownika.
- Testowanie funkcjonalności systemu jako całości oraz przeprowadzanie testów akceptacyjnych.

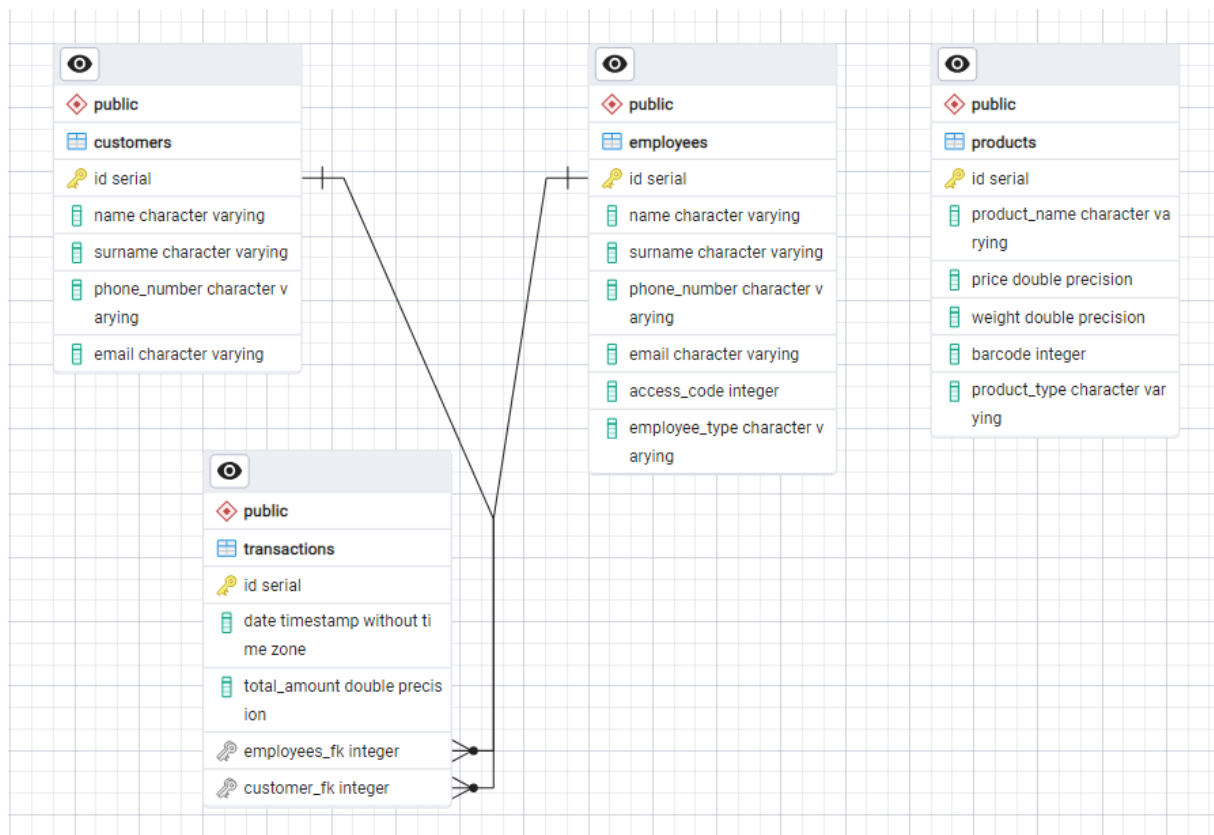
## Podział obowiązków w skrócie

- Kacper: Główne prace nad backendem i bazą danych.
- Jakub: Główne prace nad interfejsem użytkownika w Windows Forms.
- Mateusz: Prace wspomagające zarówno na backendzie, jak i z UI, oraz bazą danych.

Taki podział zadań umożliwił nam efektywne działanie i skoncentrowanie się na kluczowych elementach projektu, przy jednoczesnym wsparciu i uzupełnianiu się nawzajem w razie potrzeby.

# Baza danych

## Schemat ERD



Rysunek 2 - diagram ERD

Tabela: customers (klienci) przechowuje informacje o klientach, czyli takich którzy założyli kartę stałego klienta za pomocą formularza przy płatności. Zawiera następujące kolumny:

- id (serial): Klucz główny tabeli, automatycznie inkrementowany.
- name (character varying): Imię klienta.
- surname (character varying): Nazwisko klienta.
- phone\_number (character varying): Numer telefonu klienta.
- email (character varying): Adres email klienta.

Tabela: employees (pracownicy) przechowuje informacje o pracownikach. Zawiera następujące kolumny:

- id (serial): Klucz główny tabeli, automatycznie inkrementowany.
- name (character varying): Imię pracownika.
- surname (character varying): Nazwisko pracownika.
- phone\_number (character varying): Numer telefonu pracownika.
- email (character varying): Adres email pracownika.
- access\_code (integer): Kod dostępu pracownika, używany do autoryzacji w systemie.
- employee\_type (character varying): Typ pracownika (np. kasjer, menedżer).

Tabela: products (produkty) przechowuje informacje o produktach dostępnych w sklepie. Zawiera następujące kolumny:

- id (serial): Klucz główny tabeli, automatycznie inkrementowany.
- product\_name (character varying): Nazwa produktu.
- price (double precision): Cena produktu.
- weight (double precision): Waga produktu.
- barcode (integer): Kod kreskowy produktu.
- product\_type (character varying): Typ produktu (np. spożywczy, elektronika).

Tabela: transactions (transakcje) przechowuje informacje o transakcjach dokonanych przez klientów. Zawiera następujące kolumny:

- id (serial): Klucz główny tabeli, automatycznie inkrementowany.
- date (timestamp without time zone): Data i czas transakcji.
- total\_amount (double precision): Całkowita kwota transakcji.
- employees\_fk (integer): Klucz obcy do tabeli employees, wskazujący pracownika obsługującego transakcję.
- customer\_fk (integer): Klucz obcy do tabeli customers, wskazujący klienta dokonującego transakcji.

Relacje między tabelami

- customers (klienci) - transactions (transakcje):  
Jeden klient może mieć wiele transakcji. Relacja jest reprezentowana przez klucz obcy customer\_fk w tabeli transactions, wskazujący na kolumnę id w tabeli customers.
- employees (pracownicy) - transactions (transakcje):  
Jeden pracownik może obsługiwać wiele transakcji. Relacja jest reprezentowana przez klucz obcy employees\_fk w tabeli transactions, wskazujący na kolumnę id w tabeli employees.



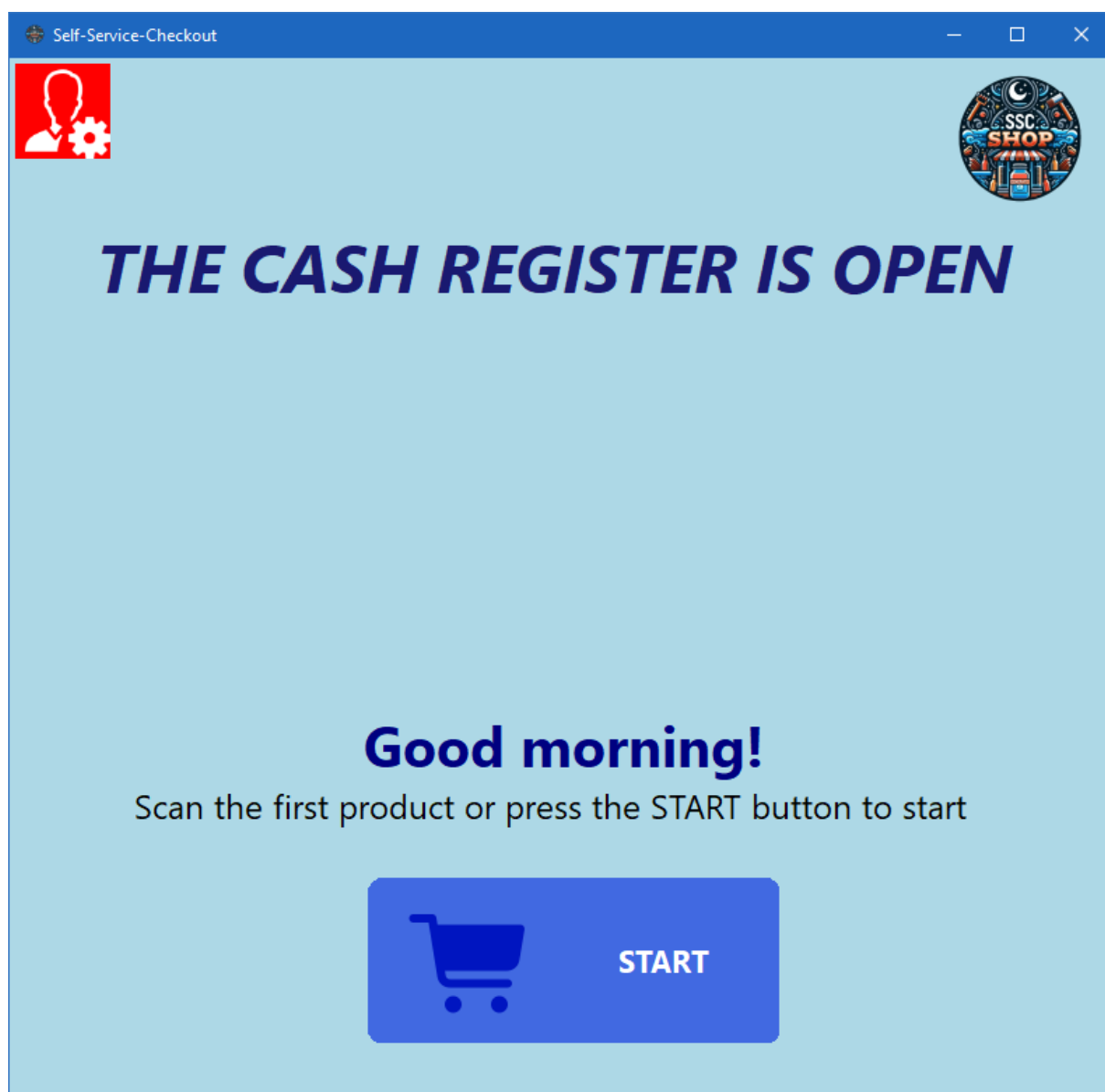
## Funkcjonalności

### Użytkownik

Użytkownicy aplikacji mają dostęp do podstawowych funkcjonalności kasy samoobsługowej, m.in. dodawanie produktów do koszyka, dodawanie wielu produktów tego samego typu do koszyka, dodawanie produktów przez wpisanie kodu kreskowego, utworzenie/podanie karty lojalnościowej, przejście do płatności.

#### 1. Ekran startowy:

Użytkownicy mogą łatwo uzyskać dostęp do panelu katalogu sklepu za pomocą jednego kliknięcia. Ten panel wyświetla powitanie w zależności od pory dnia.



Rysunek 3 - ekran powitalny

## 2. Panel główny użytkownika:

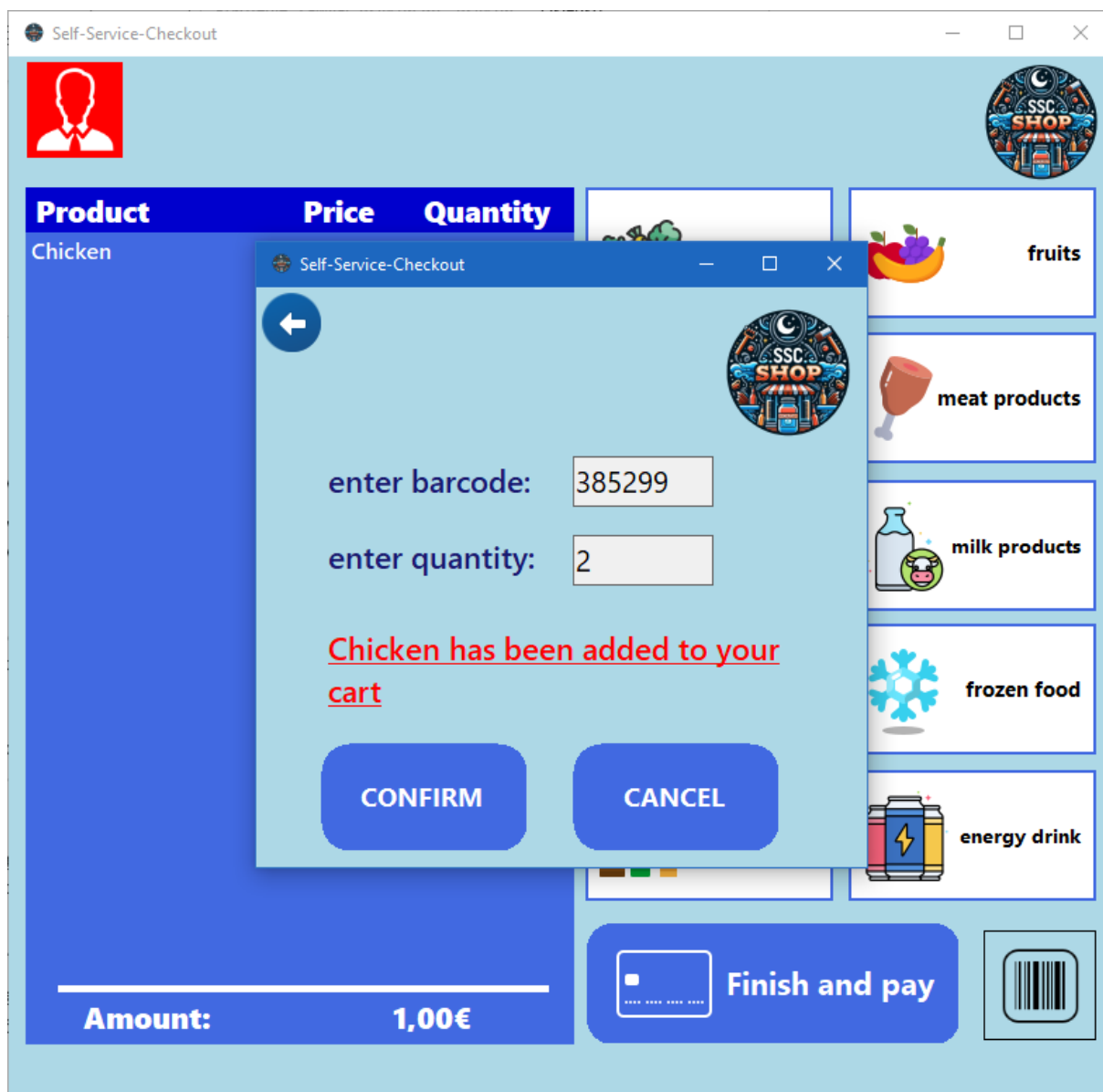
Tutaj użytkownik ma dostęp do widoku stanu koszyka, panelu kategorii produktów, komunikatów, ręcznego wprowadzenia kodu kreskowego itd.



Rysunek 4 - ekran główny

3. Skanuj kod produktu (wstaw kod produktu) / wybierz produkt z panelu listy:

Użytkownicy mogą dodać produkty do koszyka poprzez wpisanie kodu kreskowego. Alternatywnie mogą przeglądać i wybierać produkty bezpośrednio z panelu list.



Rysunek 5 - dodanie produktu do koszyka za pomocą kodu kreskowego

#### 4. Ograniczenia wiekowe

Produkty z kategorii alkohole i napoje energetyczne. Wtedy kasjer musi zatwierdzić wiek użytkownika, inaczej opcja płatności zostaje zablokowana.



Rysunek 6 - komunikat o wymogu potwierdzeniu wieku

5. Wybór ilości produktu:


Użytkownik ma dowolność w wyborze ilości obecnie zaznaczonego produktu, tak by łatwiej dodawać wiele tych samych produktów do koszyka.

Self-Service-Checkout

←


# Milk Products

Select a product from the list and then enter the quantity



Product	Price	Weight	Barcode
Milk	1	1000	789234
Yogurt	0,8	200	124356
Cheddar Cheese	8,99	300	654211
Gouda Cheese	10,5	500	712345
Butter	1,2	100	654123
Greek Yogurt	1,2	130	612367
Mozzarella Cheese	5	200	762345
Cream	0,9	200	376042

Enter quantity:



Rysunek 7 - dodanie konkretnej ilości dla wybranego produktu

## 6. Kasa:

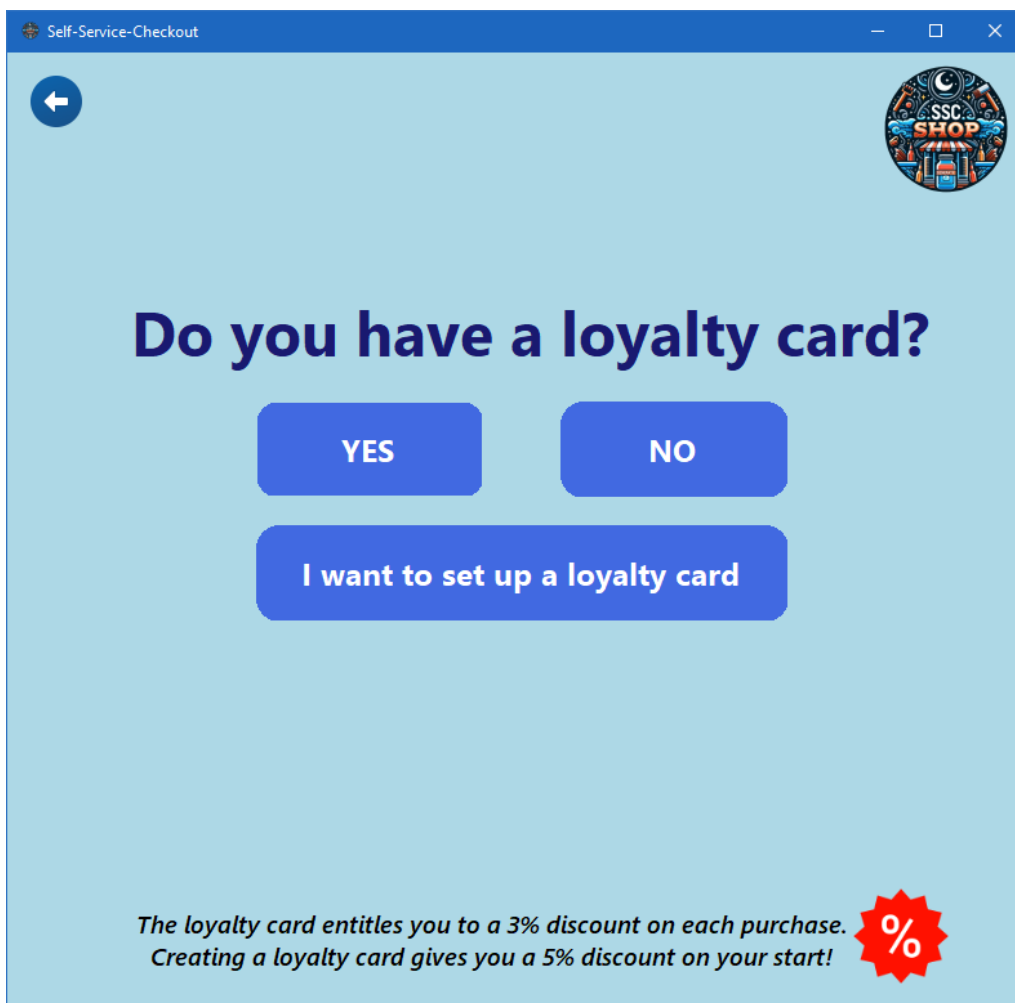
Użytkownicy mogą przejść do kasy, gdy zakończą dodawanie produktów do koszyka. Proces kasowy obejmuje wybranie metody płatności oraz ewentualne podanie/utworzenie karty lojalnościowej.



Rysunek 8 - zakończenie zakupów, przejście do płatności

## 7. Wybierz metodę płatności

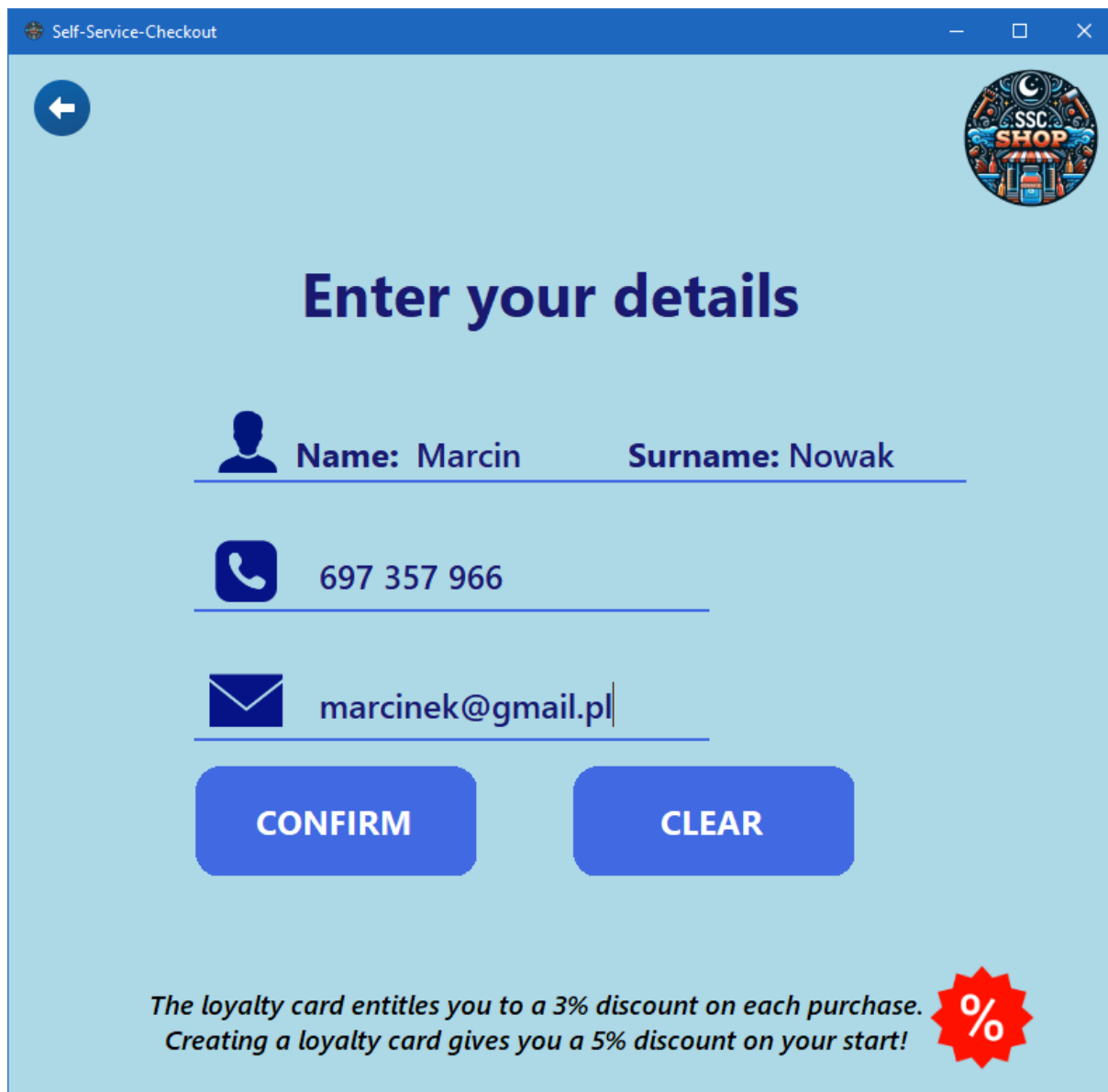
Użytkownicy mogą wybierać spośród różnych metod płatności, aby sfinalizować zakup. Dostępne opcje mogą obejmować płatności kartą debetową, płatności mobilne (Blik).



Rysunek 9 - karta stałego klienta

8. Podaj kartę lojalnościową / utwórz nową kartę:

Podczas procesu płatności użytkownicy mają możliwość podania swojej karty lojalnościowej, aby otrzymać zniżki. Nowi użytkownicy mogą również utworzyć nową kartę lojalnościową bezpośrednio z interfejsu.

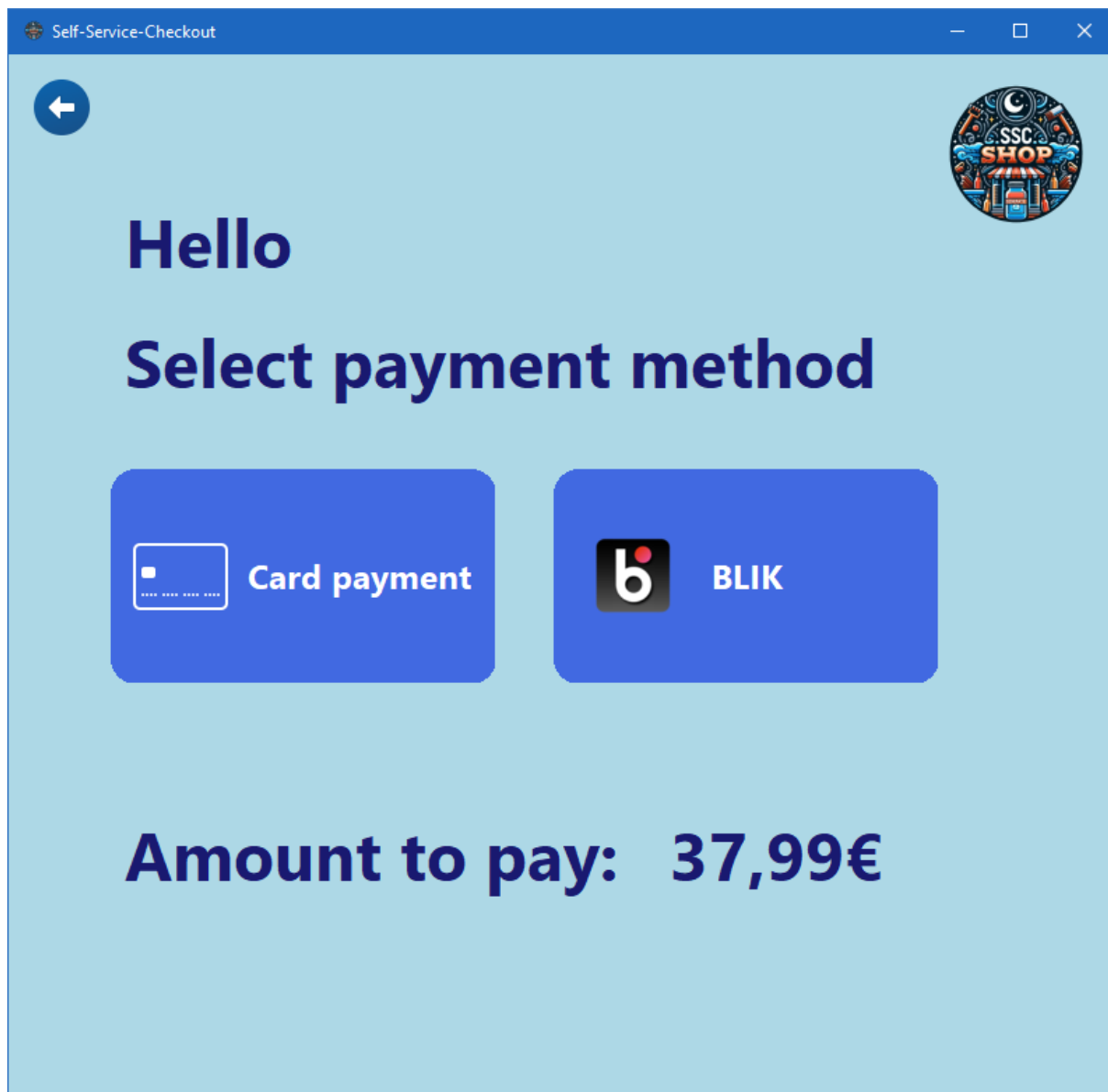


The screenshot shows a web application window titled "Self-Service-Checkout". The main heading is "Enter your details". Below this, there are three input fields, each with a blue icon on the left and a blue underline on the right. The first field is for a name, with a person icon and the text "Name: Marcin" and "Surname: Nowak". The second field is for a phone number, with a telephone icon and the text "697 357 966". The third field is for an email address, with an envelope icon and the text "marcinek@gmail.pl". Below these fields are two blue buttons: "CONFIRM" and "CLEAR". At the bottom, there is a promotional message in italics: "The loyalty card entitles you to a 3% discount on each purchase. Creating a loyalty card gives you a 5% discount on your start!" followed by a red starburst icon containing a white percentage sign.

Rysunek 10 - założenie karty stałego klienta

## 9. Wybór płatności

Użytkownik wybiera sposób płatności spośród dwóch opcji – płatność zbliżeniowa kartą lub płatność mobilna Blik. Przechodzi na okna z komunikatem o przyłożeniu karty do terminala bądź wpisania numeru płatności blik.



Rysunek 11 - wybór płatności



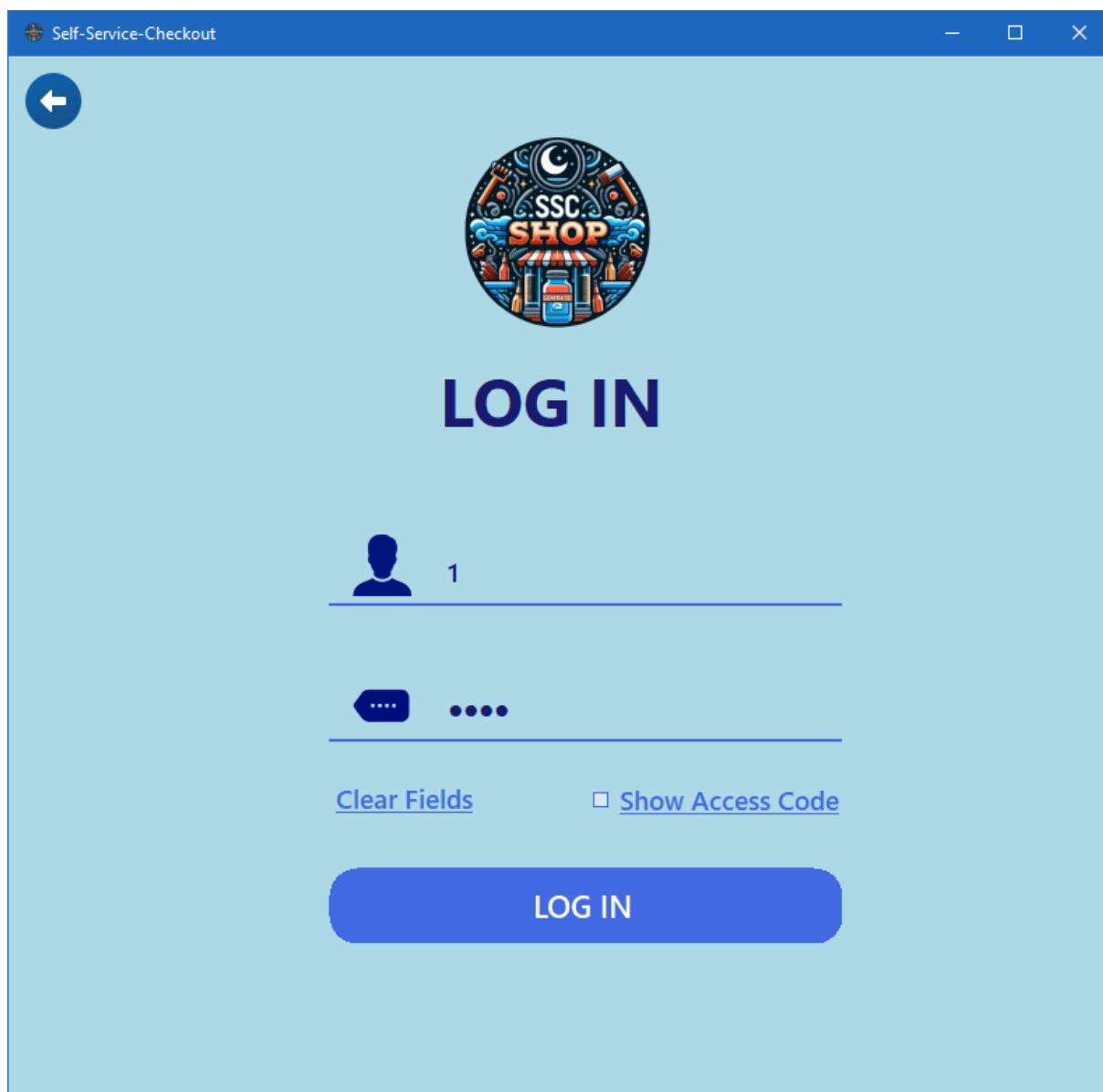
## Kasjer

Kasjer otrzymuje dodatkowe funkcje. Może zalogować się do swojego panelu poprzez panel dostępny pod guzikiem w lewym górnym rogu ekranu koszyka kasy. Podstawowe funkcje kasjera obejmują:

Weryfikacja wieku konsumenta:

1. Logowanie:

Kasjerzy mogą wprowadzić id razem ze specjalnym kodem w celu uwierzytelnienia swoich działań. Logowanie jest wymagane do wykonywania konkretnych zadań, takich jak weryfikacja wieku czy usuwanie produktów z koszyka.



Rysunek 12 - logowanie do panelu zarządzania koszykiem

## 2. Weryfikacja wieku:

Kasjerzy mogą zweryfikować wiek konsumentów przy zakupach produktów objętych ograniczeniami wiekowymi, takich jak alkohol czy napoje energetyczne. Zapewnia to zgodność z wymogami prawno-regulacyjnymi.

W przypadku niezatwierdzenia wieku, wszystkie znajdujące się produkty które są objęte weryfikacją wiekową zostają automatycznie usunięte z koszyka.

Potwierdzenie wieku odbywa się przez wybór odpowiedniego checkbox'a i zatwierdzenie operacji guzikiem potwierdzającym, w prawym dolnym rogu.

Self-Service-Checkout

←

Cart Management



Product	Price	Quantity
Baguette	1,50	8
Duck	2,50	2
Cheddar Cheese	8,99	1
Rye Bread	1,70	5
Orange	0,55	2
Carrot	0,20	12
Beer	4,99	3
Vampire Energy Drink	4,50	2

**Confirm age?**

☒ Yes

☐ No

Selling alcohol to minors  
may result in imprisonment

Product Name: \_\_\_\_\_

Quantity: \_\_\_\_\_





Rysunek 13 - zarządzanie produktami +18

3. Pomoc w usuwaniu produktu z koszyka oraz zmiana ilości produktu:

Kasjerzy mogą pomóc klientom poprzez usunięcie niechcianych produktów z ich koszyka. Funkcjonalność ta jest przydatna w przypadku błędów skanowania lub zmiany decyzji klienta.

Operacja usuwania produktów z koszyka odbywa się poprzez zaznaczenie odpowiedniego wiersza z koszyka i naciśnięcie guzika z ikoną kosza.

Zmiana ilości produktu wspierana jest przez wybór przedmiotu a następnie wprowadzenie docelowej przez kasjera ilości.

Product	Price	Quantity
Baguette	1,50	8
Duck	2,50	2
Cheddar Cheese	8,99	1
Rye Bread	1,70	5
Orange	0,55	2
Carrot	0,20	12
Beer	4,99	3
Vampire Energy Drink	4,50	2



**Confirm age?**

☐ Yes

☐ No

*Selling alcohol to minors may result in imprisonment*

Product Name: **Beer**      Quantity: **3**

Rysunek 14 - usuwanie produktów za pomocą przycisków

## Administrator

Administrator ma dostęp do funkcjonalności kasjera, ale również do operacji na bazie danych w swoim panelu.

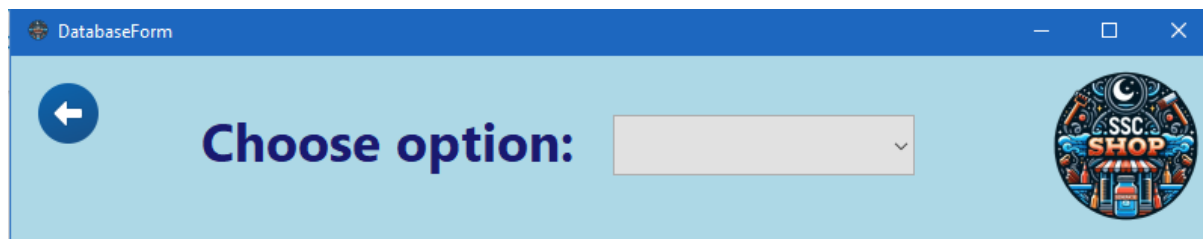
Dostęp do panelu administratora jest wyłącznie przez wejście za pomocą głównego okna i posiadania typu 'admin' w bazie, kasjerzy próbując się zalogować otrzymają odpowiedni komunikat.



Rysunek 15 - panel administracyjny

Administratorzy mogą przeprowadzać operacje CRUD na bazie danych.

#### 1. CRUD pracowników/użytkowników



Rysunek 16 - początkowy wybór tabeli do zarządzania

Dostępny po naciśnięciu odpowiedniego guzika w panelu administratora, umożliwia:

- Wyświetlanie dostępnych użytkowników/pracowników.
- Dodawanie użytkowników/pracowników, przez podane wartości w ostatnim wierszu tabeli. Po uzupełnieniu wszystkich należy wyjść z obecnie edytowanej komórki do innej, nie obecnie dodawanej komórki. W innym przypadku aplikacja doda puste pole w miejscu zaznaczonej komórki z nowego wiersza.
- Usuwanie użytkowników/pracowników poprzez wybranie wiersza i naciśnięcie klawisza „Del”.
- Edytowanie użytkowników/pracowników, poprzez podwójne kliknięcie w wybraną komórkę. Po wyjściu z komórki dane są zapisywane w bazie danych.

The screenshot shows the same 'DatabaseForm' window, but the dropdown menu now displays 'Customers'. Below the header is a table with 6 columns: 'id', 'name', 'surname', 'phone number', and 'email'. The table contains 21 rows of customer data. A red checkmark icon is visible in the bottom right corner of the application window.

	id	name	surname	phone number	email
▶	1	Bernice	McNeachtain	740 338 315	bmcneachtain0@washington...
	2	Lovell	Cordaroy	679 280 956	lcordaroy1@mail.ru
	3	Catie	Struss	936 827 963	cstruss2@elegantthemes.com
	4	Pernell	Redholls	439 211 728	predholls3@npr.org
	5	Laney	Berzins	193 512 984	lberzins4@angelfire.com
	6	Dag	Shillaber	544 955 644	dshillaber5@symantec.com
	7	Chaunce	Lauxmann	365 373 296	clauxmann6@etsy.com
	8	Kellie	Newis	869 590 488	knewis7@photobucket.com
	9	Janos	Bates	185 413 232	jbates8@statcounter.com
	10	Kirstyn	Suggey	974 103 243	ksuggey9@google.cn
	11	Yulma	Radolf	923 549 336	yradolfa@lulu.com
	12	Avigdor	Pragnall	361 948 560	apragallb@state.tx.us
	13	Rafael	Rollingson	772 222 311	rrollingsonc@symantec.com
	14	Robin	Van Salzberger	298 347 619	rvansalzbergerd@biglobe.ne.jp
	15	Isidore	Anshell	911 843 051	ianshelle@creativecommons....
	16	Deck	Bedenham	965 924 110	dbedenhamf@earthlink.net
	17	Magdalena	Smedmore	232 456 873	msmedmoreg@jimdo.com
	18	Branden	Graver	454 611 697	bgraverh@zimbio.com
	19	Hamid	Katz	631 899 111	hkatzi@tinyurl.com
	20	Maddy	Branney	176 616 704	mbranneyj@nifty.com
	21	Andonis	Imlach	816 238 696	aimlachk@aol.com

Rysunek 17 - zarządzanie wybraną tabelą

## 2. CRUD produktów

Po wejściu w product management pojawia się intuicyjny formularz edycji, dodawania, oraz usuwania danych produktów. Operacje tutaj wykonywane odbywają się za pomocą funkcji pgSQL. Dostępne tutaj operacje obejmują:

- Wyszukiwanie produktu po jego id
- Dodawanie nowego produktu w formularzu i naciśnięciu guzika „Add product”
- Edycje istniejącego produktu poprzez naciśnięcie jego id z widoku i edycję w formularzu a następnie naciśnięcie guzika „Save”.
- Odświeżenie widoku przez guzik „Refresh”.
- Usunięcie zaznaczonego przedmiotu przez guzik „Delete”.

The screenshot shows a web application titled "Self-Service-Checkout". At the top, there is a search bar with the label "id" and a blue "Search" button. To the right of the search bar is a circular logo with the text "SSC SHOP". Below the search bar is a table with the following data:

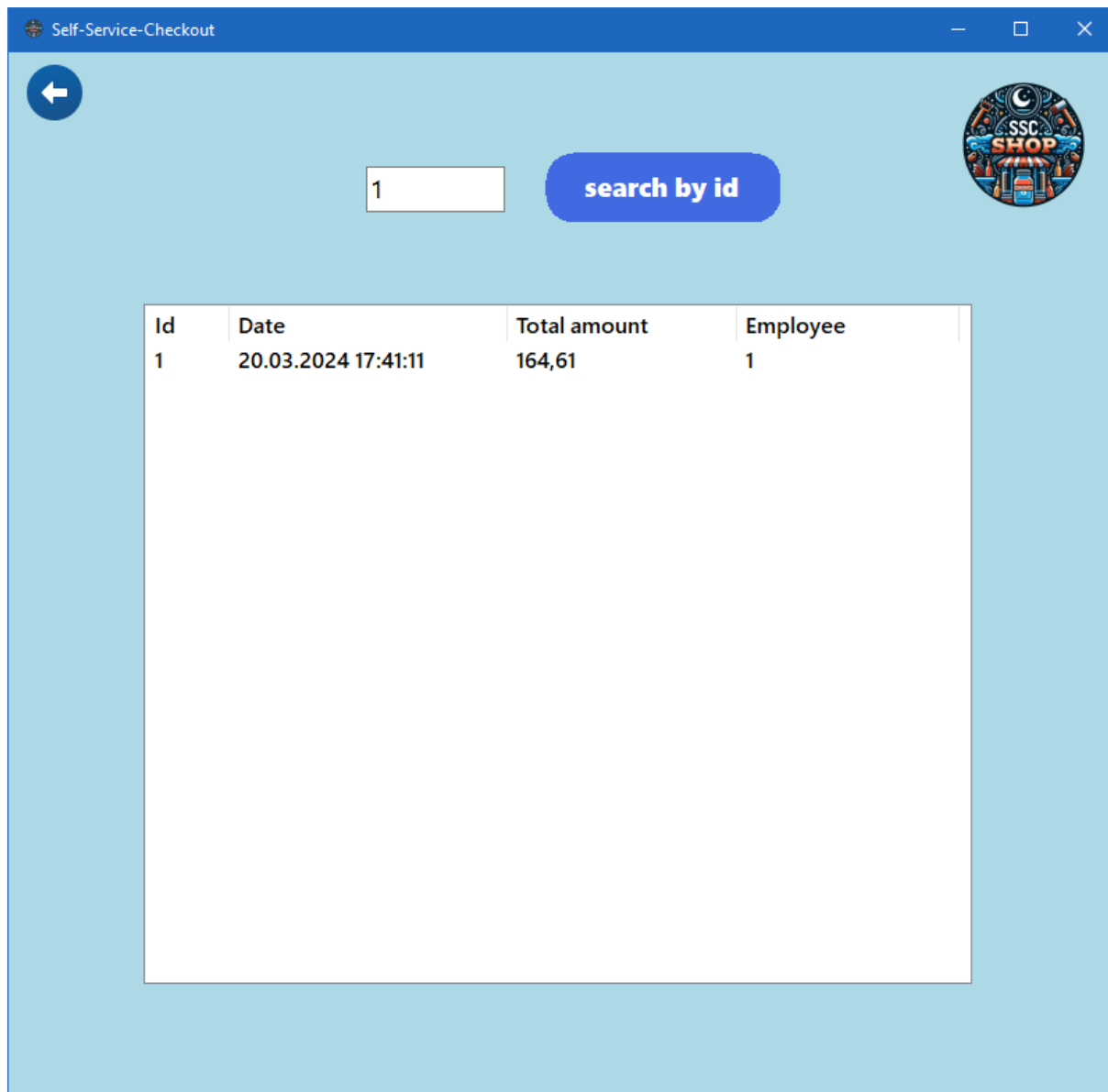
Id	Product name	Price	Weight	Barcode	type
1	Apple	0,45	150,00	575448	fruit
2	Banana	0,25	120,00	633119	fruit
3	Orange	0,55	200,00	800798	fruit
4	Strawberry	1,80	100,00	457303	fruit
5	Grapes	3,00	500,00	905518	fruit
6	Watermelon	4,50	5000,00	870407	fruit
7	Pineapple	2,00	1500,00	354329	fruit
8	Peach	0,70	180,00	823453	fruit
9	Pear	0,65	180,00	868494	fruit
10	Mango	1,30	300,00	893834	fruit
11	Kiwi	0,60	100,00	892871	fruit
12	Plum	0,50	80,00	922119	fruit
13	Raspberry	3,50	200,00	185935	fruit
14	Blueberry	3,00	150,00	997642	fruit
15	Blackberry	2,00	100,00	625204	fruit
16	Cherry	1,00	100,00	176228	fruit
17	Apricot	0,90	150,00	934182	fruit
18	Papaya	1,80	1000,00	421294	fruit
19	Guawa	1,30	250,00	546329	fruit
20	Lychee	3,50	250,00	859806	fruit
21	Carrot	0,20	100,00	659736	vegetable

Below the table, there are input fields for "Price", "Weight", and "Barcode", along with a dropdown menu. To the right of these fields is a link "Clear Fields". At the bottom of the interface, there are four buttons: "Refresh" (blue), "Save" (blue), "Add Product" (green), and "Delete" (red).

Rysunek 18 - widok zarządzania produktami

### 3. Wyświetlanie transakcji dla podanych klientów

Administrator wyświetla historię transakcji dla wybranego użytkownika. Wpisuje numer id użytkownika a następnie za pomocą przycisku potwierdza swój wybór. Pole to sprawdza czy istnieje klient o podanym id, bądź czy został wprowadzony właściwy format, w przypadku niepowodzenia wyświetla stosowny komunikat.



The screenshot shows a web application window titled "Self-Service-Checkout". The interface has a light blue background. At the top left is a back arrow button. At the top right is a circular logo with the text "SSC SHOP" and a shopping cart icon. In the center, there is a text input field containing the number "1" and a blue button labeled "search by id". Below this, a table displays transaction history. The table has four columns: "Id", "Date", "Total amount", and "Employee". The first row shows the transaction details for ID 1.

Id	Date	Total amount	Employee
1	20.03.2024 17:41:11	164,61	1

Rysunek 19 - wyświetlanie historii transakcji

## Wykorzystanie języka proceduralnego

W naszym projekcie wykorzystaliśmy język proceduralny PL/pgSQL w bazie danych PostgreSQL do tworzenia funkcji, procedur, triggerów oraz kursorów. Poniżej znajduje się opis poszczególnych funkcjonalności, które zostały zaimplementowane przy użyciu tego języka.

### Procedury i funkcje

#### 1. **get\_all\_products:**

Funkcja zwracająca wszystkie produkty z tabeli products. Jest używana do pobierania pełnej listy produktów, co umożliwia wyświetlanie ich w interfejsie użytkownika.

```
CREATE OR REPLACE FUNCTION get_all_products()
RETURNS TABLE (
    id INTEGER,
    product_name VARCHAR,
    price DOUBLE PRECISION,
    weight DOUBLE PRECISION,
    barcode INTEGER,
    product_type VARCHAR
) AS $$
BEGIN
    RETURN QUERY SELECT * FROM products
    ORDER BY id;
END;
$$ LANGUAGE plpgsql;
```

#### 2. **add\_product:**

Funkcja dodająca nowy produkt do tabeli products. Przyjmuje jako parametry nazwę produktu, cenę, wagę, kod kreskowy oraz typ produktu. Umożliwia dodawanie nowych pozycji do bazy danych z poziomu aplikacji.

```
CREATE OR REPLACE FUNCTION add_product(
    _product_name VARCHAR,
    _price DOUBLE PRECISION,
    _weight DOUBLE PRECISION,
    _barcode INTEGER,
    _product_type VARCHAR
)
RETURNS VOID AS $$
BEGIN
    INSERT INTO products (product_name, price, weight, barcode, product_type)
    VALUES (_product_name, _price, _weight, _barcode, _product_type);
END;
$$ LANGUAGE plpgsql;
```



### 3. **update\_product:**

Funkcja aktualizująca istniejący produkt w tabeli products na podstawie przekazanego identyfikatora. Aktualizuje takie pola jak nazwa, cena, waga, kod kreskowy i typ produktu. Jest wykorzystywana do edytowania szczegółów istniejących produktów.

```
CREATE OR REPLACE FUNCTION update_product(  
    _id INTEGER,  
    _product_name VARCHAR,  
    _price DOUBLE PRECISION,  
    _weight DOUBLE PRECISION,  
    _barcode INTEGER,  
    _product_type VARCHAR  
)  
RETURNS VOID AS $$  
BEGIN  
    UPDATE products  
    SET product_name = _product_name,  
        price = _price,  
        weight = _weight,  
        barcode = _barcode,  
        product_type = _product_type  
    WHERE id = _id;  
END;  
$$ LANGUAGE plpgsql;
```

### 4. **delete\_product:**

Procedura usuwająca produkt z tabeli products na podstawie identyfikatora. Umożliwia usuwanie produktów, które są już niepotrzebne lub błędnie wprowadzone.

```
CREATE OR REPLACE PROCEDURE delete_product(_id INTEGER)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    DELETE FROM products WHERE id = _id;  
END;  
$$;
```

## 5. prod\_stamp:

Funkcja walidująca dane produktu przed ich wstawieniem lub aktualizacją. Sprawdza, czy wszystkie wymagane pola są wypełnione i mają odpowiednie wartości (np. cena nie może być ujemna, kod kreskowy musi mieć 6 cyfr). Jest wywoływana przez trigger przed każdą operacją wstawiania lub aktualizacji, aby zapewnić integralność danych.

```
CREATE FUNCTION prod_stamp() RETURNS trigger AS $prod_stamp$
BEGIN
    IF NEW.product_name IS NULL OR NEW.product_name = '' THEN
        RAISE EXCEPTION 'product name cannot be null';
    END IF;
    IF NEW.price IS NULL THEN
        RAISE EXCEPTION '% cannot have null price', NEW.product_name;
    END IF;
    IF NEW.price < 0 THEN
        RAISE EXCEPTION '% cannot have a negative price', NEW.product_name;
    END IF;
    IF NEW.weight IS NULL THEN
        RAISE EXCEPTION '% cannot have null weight', NEW.product_name;
    END IF;
    IF NEW.weight < 0 THEN
        RAISE EXCEPTION '% cannot have a negative weight', NEW.product_name;
    END IF;
    IF NEW.barcode < 100000 OR NEW.barcode > 999999 THEN
        RAISE EXCEPTION '% must have a 6-digit barcode', NEW.product_name;
    END IF;
    IF EXISTS (SELECT 1 FROM products WHERE barcode = NEW.barcode AND id !=
NEW.id) THEN
        RAISE EXCEPTION 'Barcode % already exists', NEW.barcode;
    END IF;
    IF NEW.barcode IS NULL THEN
        RAISE EXCEPTION '% cannot have null barcode', NEW.product_name;
    END IF;
    IF NEW.product_type IS NULL OR NEW.product_type = '' THEN
        RAISE EXCEPTION '% cannot have null product type', NEW.product_name;
    END IF;
    RETURN NEW;
END;
$prod_stamp$ LANGUAGE plpgsql;

CREATE TRIGGER prod_stamp BEFORE INSERT OR UPDATE ON products
FOR EACH ROW EXECUTE FUNCTION prod_stamp();
```

## 6. `get_product_by_id`:

Funkcja wykorzystująca kursor do pobrania szczegółowych informacji o produkcie na podstawie jego identyfikatora. Umożliwia wyszukiwanie i wyświetlanie danych konkretnego produktu.

```
CREATE OR REPLACE FUNCTION get_product_by_id(product_id INTEGER)
RETURNS TABLE (
    id INTEGER,
    product_name VARCHAR,
    price DOUBLE PRECISION,
    weight DOUBLE PRECISION,
    barcode INTEGER,
    product_type VARCHAR
) AS $$
DECLARE
    product_cursor CURSOR FOR
        SELECT
            p.id AS product_id,
            p.product_name,
            p.price,
            p.weight,
            p.barcode,
            p.product_type
        FROM products p
        WHERE p.id = product_id;
    product_record RECORD;
BEGIN
    OPEN product_cursor;

    LOOP
        FETCH product_cursor INTO product_record;
        EXIT WHEN NOT FOUND;

        id := product_record.product_id;
        product_name := product_record.product_name;
        price := product_record.price;
        weight := product_record.weight;
        barcode := product_record.barcode;
        product_type := product_record.product_type;

        RETURN NEXT;
    END LOOP;

    CLOSE product_cursor;
END;
$$ LANGUAGE plpgsql;
```

## 7. `get_transactions_by_customer`:

Funkcja wykorzystująca kursor do pobrania wszystkich transakcji powiązanych z danym klientem na podstawie jego identyfikatora. Pozwala na przeglądanie historii zakupów klienta.

```
CREATE OR REPLACE FUNCTION get_transactions_by_customer(customer_id INTEGER)
RETURNS TABLE (
    id INTEGER,
    date TIMESTAMP,
    total_amount DOUBLE PRECISION,
    employees_fk INTEGER,
    customer_fk INTEGER
) AS $$
DECLARE
    transaction_cursor CURSOR FOR
        SELECT
            t.id,
            t.date,
            t.total_amount,
            t.employees_fk,
            t.customer_fk
        FROM transactions t
        WHERE t.customer_fk = customer_id;
    transaction_record RECORD;
BEGIN
    OPEN transaction_cursor;

    LOOP
        FETCH transaction_cursor INTO transaction_record;
        EXIT WHEN NOT FOUND;

        id := transaction_record.id;
        date := transaction_record.date;
        total_amount := transaction_record.total_amount;
        employees_fk := transaction_record.employees_fk;
        customer_fk := transaction_record.customer_fk;

        RETURN NEXT;
    END LOOP;

    CLOSE transaction_cursor;
END;
$$ LANGUAGE plpgsql;
```

## Integracja z aplikacją

Wszystkie powyższe funkcje i procedury zostały zintegrowane z kontekstem bazy danych `SscdbContext`. Dzięki temu mogliśmy w łatwy sposób wywoływać operacje bazodanowe z poziomu aplikacji, korzystając z metod asynchronicznych, takich jak:

- `GetAllProductsAsync` – pobieranie wszystkich produktów,
- `AddProductAsync` – dodawanie nowego produktu,
- `UpdateProductAsync` – aktualizowanie danych produktu,
- `DeleteProductAsync` – usuwanie produktu,
- `GetProductByIdAsync` – pobieranie szczegółowych informacji o produkcie na podstawie identyfikatora,
- `GetTransactionByIdAsync` – pobieranie transakcji powiązanych z klientem.

Dzięki wykorzystaniu PL/pgSQL i integracji z C#, nasza aplikacja mogła sprawnie i efektywnie zarządzać danymi w bazie danych, zapewniając jednocześnie integralność i poprawność danych.

## Podsumowanie

Projekt polegał na stworzeniu aplikacji wspomagającej samoobsługowe kasy fiskalne, wykorzystującej C# z Windows Forms oraz PostgreSQL z PL/pgSQL.

### Kluczowe funkcjonalności

#### 1. Interfejs użytkownika:

Czytelny interfejs użytkownika zaimplementowany w Windows Forms, integracja z bazą danych za pomocą ORM, walidacja danych wejściowych.

#### 2. Baza danych:

Projektowanie struktury, tworzenie i optymalizacja procedur oraz funkcji w PL/pgSQL.

#### 3. Logika biznesowa:

Operacje typowej kasy fiskalnej, udoskonalone o zarządzanie klientami, pracownikami, produktami i podglądem transakcji.

### Wnioski

Realizacja projektu umożliwiła zdobycie doświadczeń w pracy z technologiami C#, Windows Forms i PL/pgSQL. Dzięki efektywnej współpracy i podziałowi obowiązków, zespół stworzył funkcjonalną i intuicyjną aplikację do zarządzania samoobsługowymi kasami fiskalnymi.