Restaurant manager

Konsolowa aplikacja zarządzająca restauracją, obsługująca zamówienia, pracowników, klientów i menu.

Dominika Bomba, Nikodem Jokiel

Opis projektu

- **Cel projektu:** Stworzenie funkcjonalnej aplikacji konsolowej, która umożliwia zarządzanie procesami restauracyjnymi.
- **Co robi aplikacja:**
  - Pozwala na:
    - przegląd menu, wybór dań i złożenie zamówień przez klienta,
    - obsługę zamówień przez kucharzy i kelnerów,
    - zarządzanie rolami użytkowników przez administratora.
    - Przegląd przychodów restauracji przez administratora
    - zarządzanie menu restauracji
- **Dla kogo jest przeznaczona:** Dla właścicieli, pracowników restauracji oraz klientów

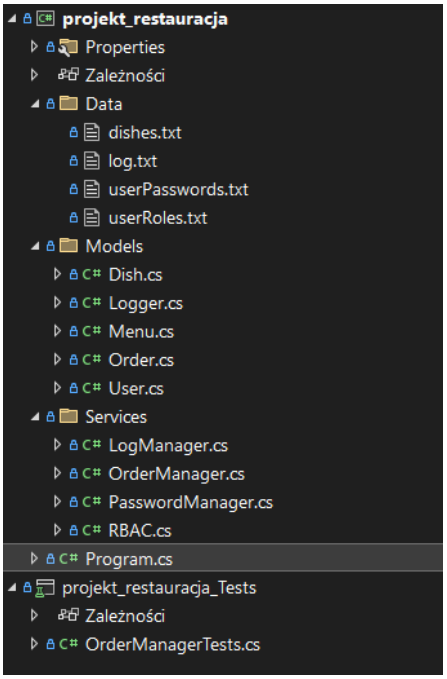
Technologie

- **Język programowania:** C#
- **Środowisko:** .NET 8.0
- **IDE:** Visual Studio 2022
- **Biblioteki:**
  - Spectre.Console (do obsługi interfejsu w konsoli)
  - XUnit (Do testów jednostkowych)

Struktura katalogów

Główny folder:

- **Program.cs** – plik główny uruchamiający aplikację
- **Models**/ – zawiera klasy używane w programie
- **Services**/ – klasy obsługujące logikę aplikacji
  - **LogManager**
  - **OrderManager**
  - **PasswordManager**
  - **RBAC**
- **Data**/ – zawiera pliki tekstowe w których przechowywane są dane
  - log.txt - logi,
  - dishes.txt - menu,
  - userPasswords.txt - użytkownicy z hasłami,
  - userRoles.txt - użytkownicy z rolami



Folder z testami jednostkowymi

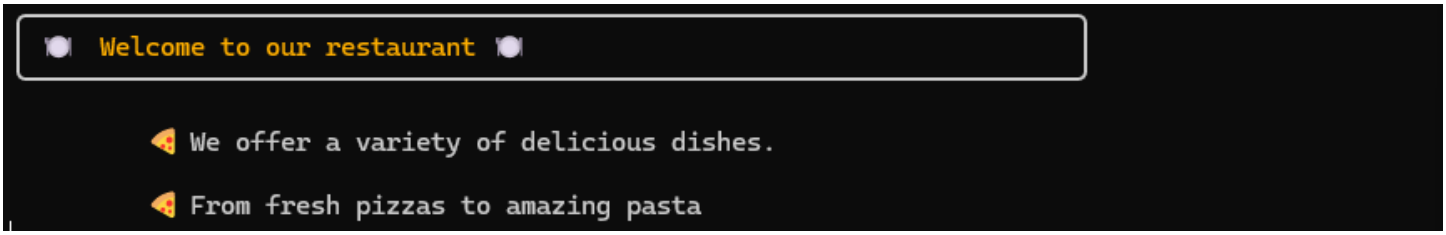
Instrukcja instalacji i uruchomienia

- **Wymagania systemowe:**
  - **.NET 8.0** lub nowszy
  - System operacyjny**: Windows/Linux/macOS**
  - IDE: **Visual Studio 2022**
  - Konsola **PowerShell**!
    - Otwórz ustawienia systemowe (Naciśnij Windows + I)
    - Przejdź do sekcji "Aktualizacje i zabezpieczenia" → Dla deweloperów.
    - W sekcji Tryb dewelopera wybierz Włącz tryb dewelopera.
    - Uruchom PowerShell w trybie deweloperskim. Kliknij prawym przyciskiem myszy na przycisk Start i wybierz Windows PowerShell (Administrator).
- **Klonowanie repozytorium/pobranie:**
  - w konsoli VS 2022 wpisać: git clone https://github.com/DominikaBomba/obiektowe_2025
  - lub rozpakować plik .zip z mooodle
- **Uruchomienie w konsoli:**
  - cd [directory]\obiektowe_2025\projekt_restauracja\projekt_restauracja
  - dotnet build
  - dotnet run
- **Uruchomienie w Visual Studio 2022:**

Opis działania aplikacji

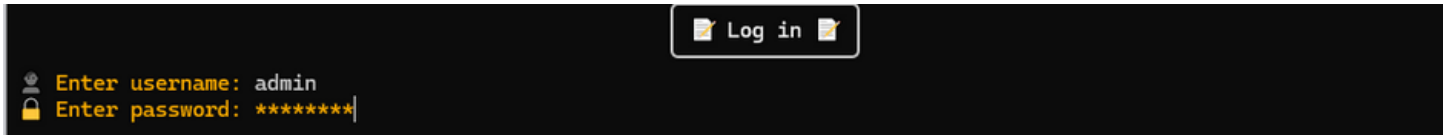
- **Funkcjonalności:**
  - **Rejestracja i logowanie**
  - Logowanie z rolami: admin, klient, kucharz, kelner
  - Składanie i obsługa zamówień
  - Zmiana statusu zamówienia (Placed > Cooked > Served > Paid)
  - Wyświetlanie zamówień z ostatnimi zmianami (dla admina)
  - Zarządzanie menu
- **Przepływ programu:**
  - Logowanie > Menu główne zależne od roli > Działania
- **Dane przetwarzane:**
  - Informacje o użytkownikach i ich rolach
  - Lista zamówień i statusy
  - Menu i dostępne dania

Strona główna

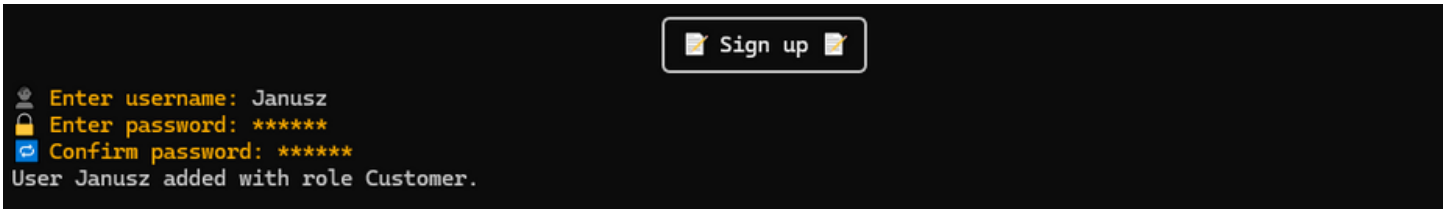


Menu z wyborem:

- logowania



- rejestracji



Klient

- ma możliwość **wyświetlenia menu**
- ma możliwość **złożenia zamówienia** - składającego się z wielu dań
- może **sprawdzić status** swoich zamówień
- może **zapłacić** za zamówienie

```
Select an option:

> Menu
  Orders
  Log out
  Exit the program
```

Option	Description
1	View a full menu
2	Back to main menu

Option	Description
1	Place an Order
2	Check My Orders
3	Pay for Order
4	Back to main menu

```
Logged in as:  customer
Order with id 4 ready to eat
```

Powiadomienia o gotowych daniach do jedzenia!

Funkcje

Category	Dish	Price (PLN)
1. Appetizer		
	MozzarellaSticks	12,00 zł
	Guacamole	9,50 zł
	MozzarellaSticks	17,00 zł
	Nazwa	12,10 zł
2. Main		
	Spaghetti	6,00 zł
	Spaghetti	13,00 zł
	Pizza	17,90 zł
	Steak	14,00 zł
	Burger	12,00 zł
	Spaghetti	17,00 zł
	Burger	15,00 zł
	Lasagna	9,99 zł
	Pasta	21,22 zł
	Danie1	11,22 zł
	Pierogi	19,99 zł
	kaczka	40,00 zł
3. Dessert		
	PannaCotta	11,10 zł
	PannaCotta	12,00 zł
4. Beverage		
	Coffee	7,00 zł
	Lemonade	19,00 zł

Wyświetlanie **menu**

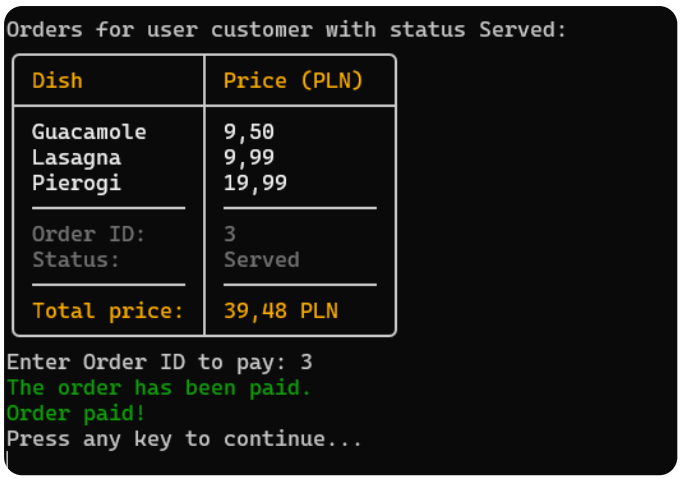
```
Dish kaczka added to your order.

What would you like to do?

> Add dish to your order
  Place an order
```

Dodanie dania do zamówienia

Your final order:

Dish	Price (PLN)
PannaCotta	11,10
kaczka	40,00
Order ID:	2
Status:	Placed
Total price:	51,10 PLN

Złożenie zamówienia



Kucharz gotuje zamówienie

↓

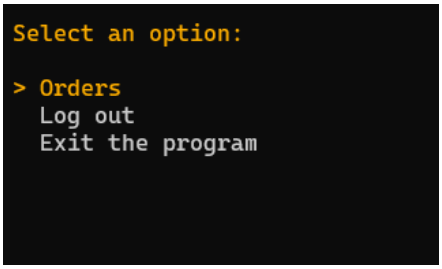
zostaje podane przez kelnera

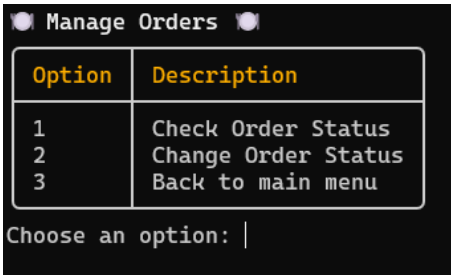
↓

zostaje opłacone przez klienta

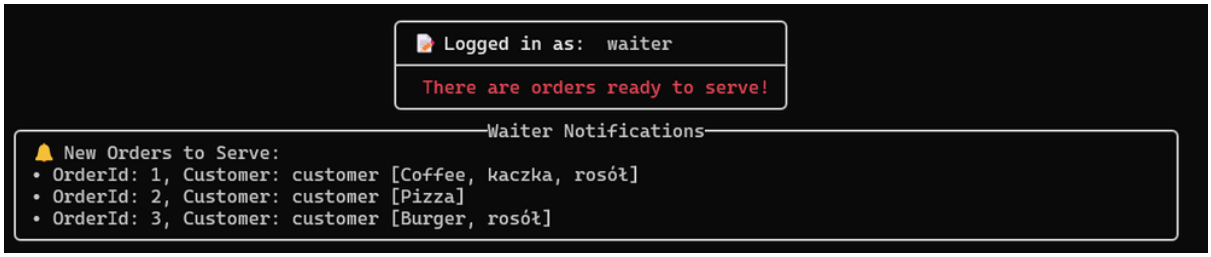
Kelner

- ma możliwość sprawdzania zamówień które są ugotowane
- ma możliwość zaserwowania zamówienia

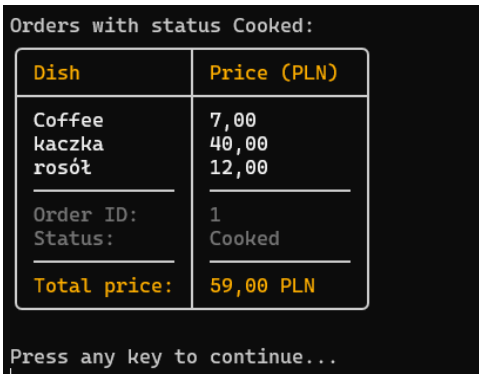




Funkcje



Powiadomienia o zamówieniach do podania

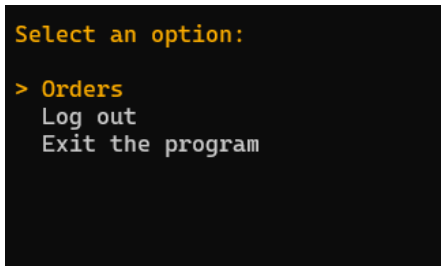


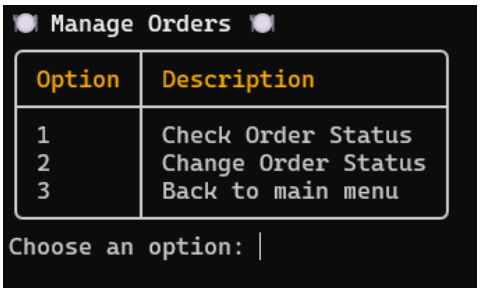
Wyświetlanie zamówień gotowych do podania

Serwowanie zamówień

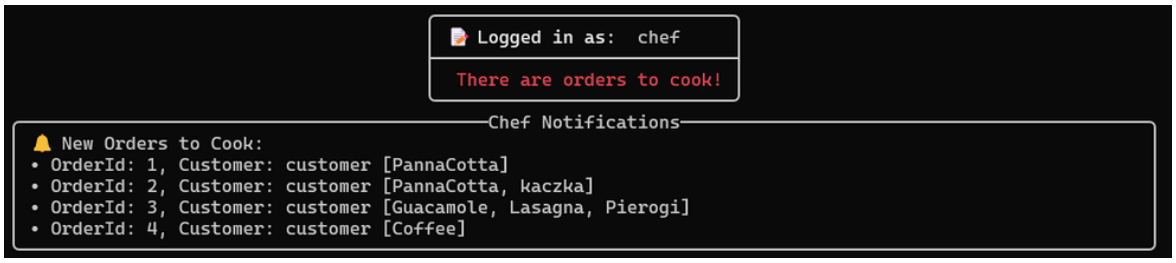
Kucharz

- ma możliwość sprawdzania zamówień które są złożone przez klientów
- ma możliwość ugotowania zamówienia

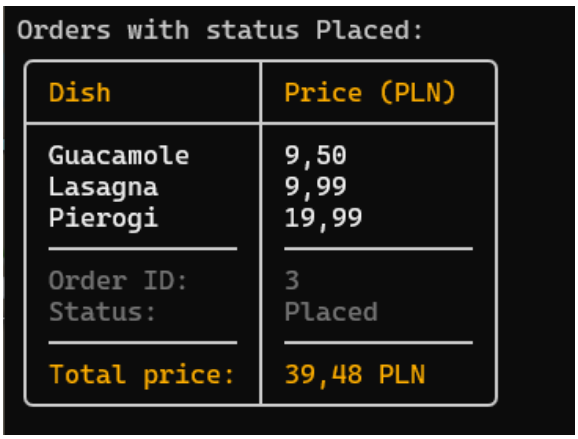




Funkcje



Powiadomienia o zamówieniach do ugotowania

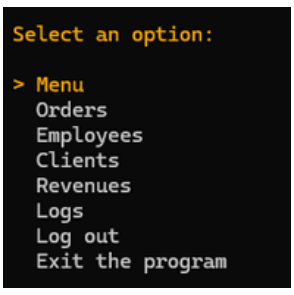


Wyświetlanie złożonych zamówień

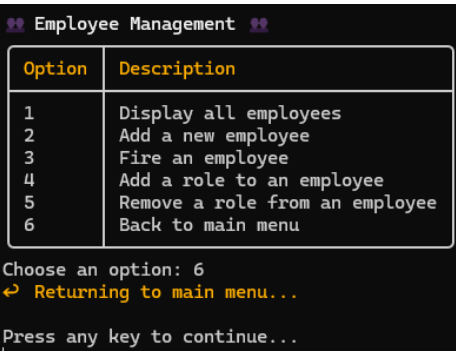
Gotowanie zamówień

Administrator

- ma dostęp do **edycji opcji w menu**
- widzi wszystkie **zamówienia**
- **zarządza pracownikami**
- **zarządza klientami**
- może **wyświetlać wszystkie zarobki** restauracji
- może **wyświetlać logi**

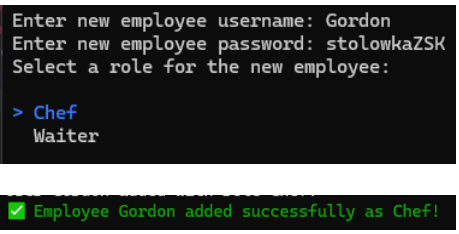


Przykładowe funkcje Administratora

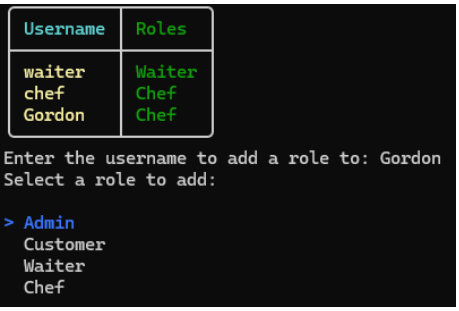


zarządzanie pracownikami

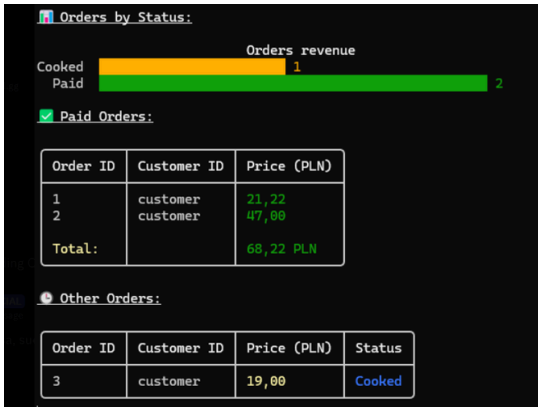
zwalnianie pracowników



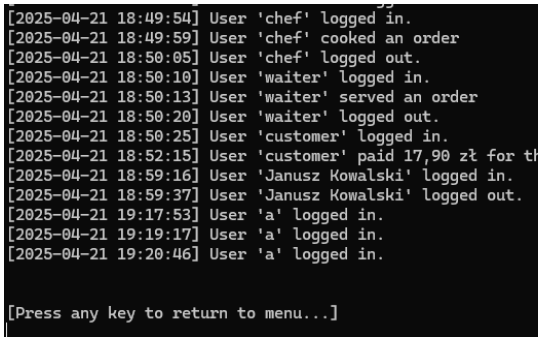
dodawanie pracowników



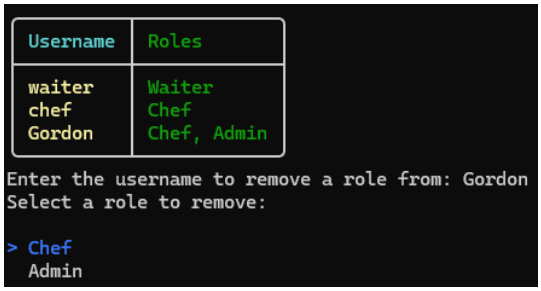
dodawanie roli użytkownikom



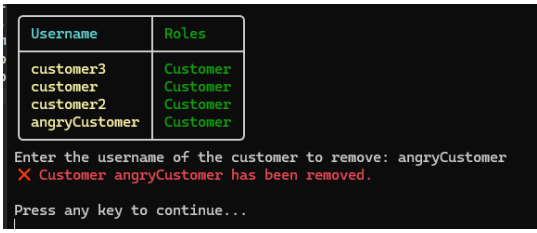
przeglądanie przychodów oraz zamówień



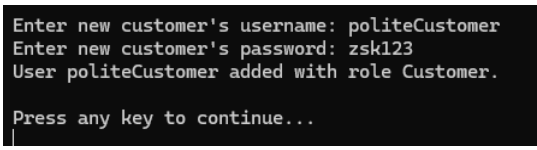
przeglądanie logów



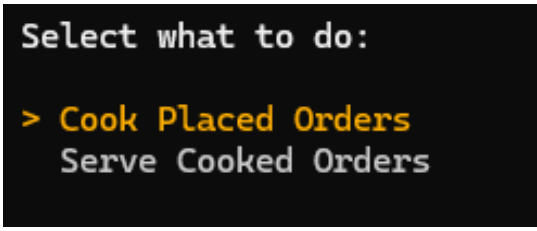
usuwanie roli użytkownikom



usuwanie klientów



dodawanie klientów



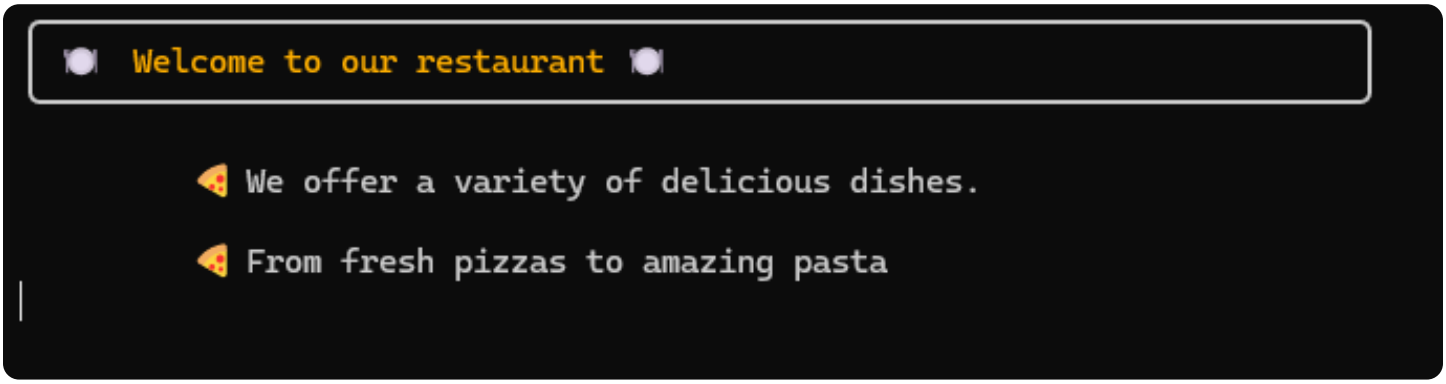
zmienianie statusu zamówienia

Przykłady użycia

Jak korzystać z aplikacji? – instrukcja krok po kroku

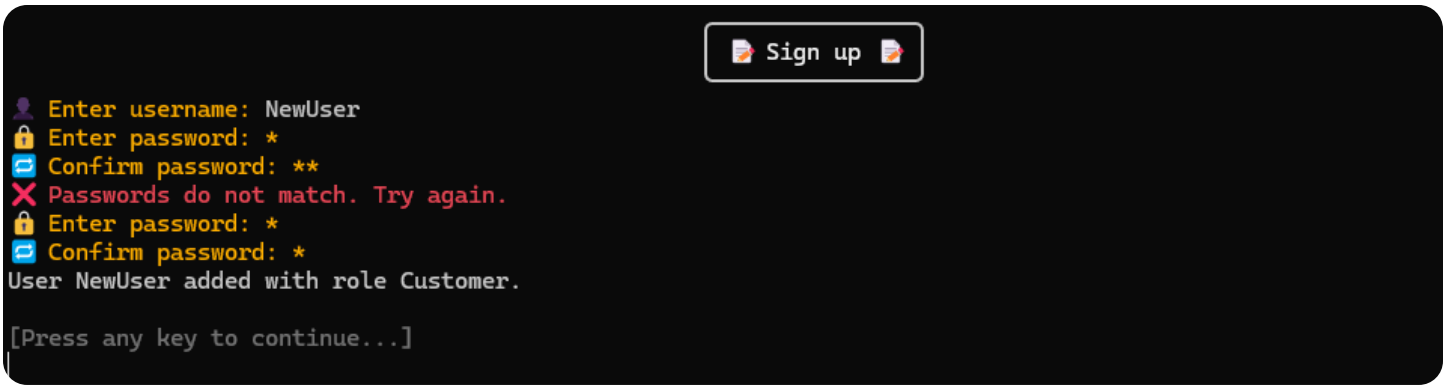
1. Uruchomienie aplikacji

- Otwórz projekt w Visual Studio.
- Uruchom aplikację (np. za pomocą skrótu Ctrl + F5).
- Po uruchomieniu pojawi się ekran logowania.



2. Logowanie/Rejestracja

- Wprowadź nazwę użytkownika i hasło.
- W zależności od przypisanej roli, zostaną wyświetlone odpowiednie opcje w menu.



Admin – zarządzanie całym systemem.

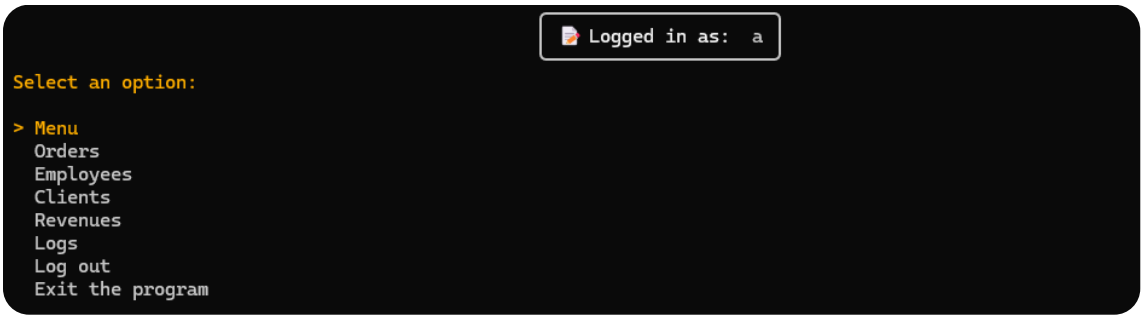
Chef – kucharz, zmienia statusy zamówień na „ugotowane”.

Waiter – kelner, zmienia statusy zamówień na „podane” i „zapłacone”.

Customer – klient, może składać zamówienia.

3. Menu główne po zalogowaniu

Po zalogowaniu zostanie wyświetlone menu z akcjami przypisanymi do Twojej roli.



DANE DO LOGOWANIA

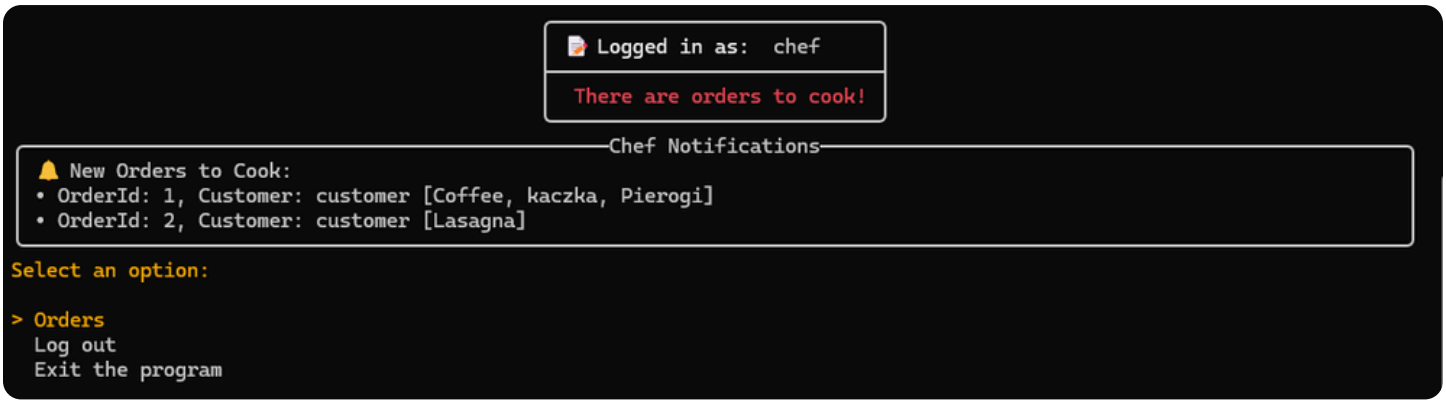
Username	Password
customer	password
chef	password
waiter	password
admin	password

4. Składanie zamówienia (rola klienta)

1. Zaloguj się jako użytkownik z rolą Customer
   - dostępne dane do zalogowania:
     - username: customer
     - password: password
2. Przeglądaj dostępne dania.
3. Wybierz pozycje i złóż zamówienie.
4. Zamówienie zostanie zapisane ze statusem „placed”.

5. Obsługa zamówienia

1. Chef loguje się i przegląda listę zamówień.
2. Zmienia status wybranego zamówienia na **Cooked**.
3. Waiter loguje się i zmienia status na **Served**.
4. Klient się loguje się i może zmienić status podanego zamówienia na Zapłacone (**Paid**)

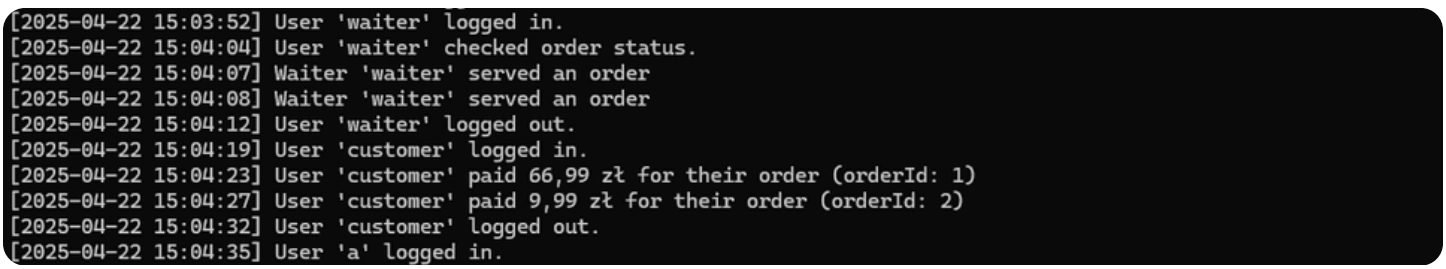


6. Przegląd przychodów

- Administrator może wybrać opcję wyświetlenia przychodów.
- Do przychodu wliczają się tylko zamówienia, które osiągnęły status „Zapłacone”.

7. Logi operacji

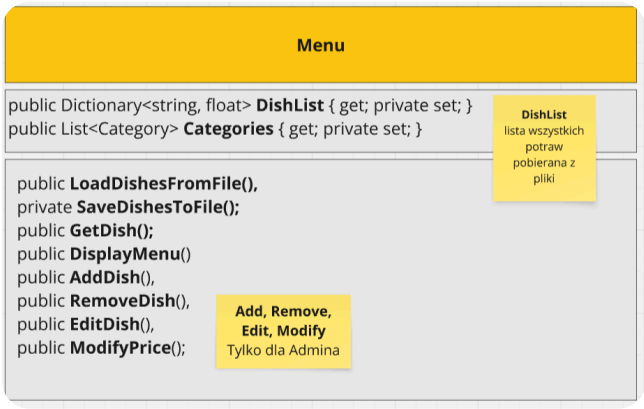
- Działania takie jak: dodanie użytkownika, zmiana statusu, itp, są zapisywane do pliku log.txt.
- Logi może wyświetlić admin



Struktury danych i klasy

Klasa Menu

zarządza listą dań i kategorii w menu restauracji. Umożliwia ładowanie dań z pliku, dodawanie, usuwanie, modyfikowanie cen dań, oraz wyświetlanie menu w formie tabeli. Dane są przechowywane w pliku, a zmiany są zapisywane po każdej modyfikacji.



Struktura pliku dishes.txt - z którego ładowane jest menu

```
11.1,Dessert,PannaCotta
7,Beverage,Coffee
12,Appetizer,MozzarellaSticks
6,Main,Spaghetti
13,Main,Spaghetti
12,Dessert,PannaCotta
17.9,Main,Pizza
14,Main,Steak
12,Main,Burger
9.5,Appetizer,Guacamole
17,Main,Spaghetti
15,Main,Burger
17,Appetizer,MozzarellaSticks
9.99,Main,Lasagna
19,Beverage,Lemonade
21.22,Main,Pasta
11.22,Main,Danie1
19.99,Main,Pierogi
12.1,Appetizer,Nazwa
40,Main,kaczka
```

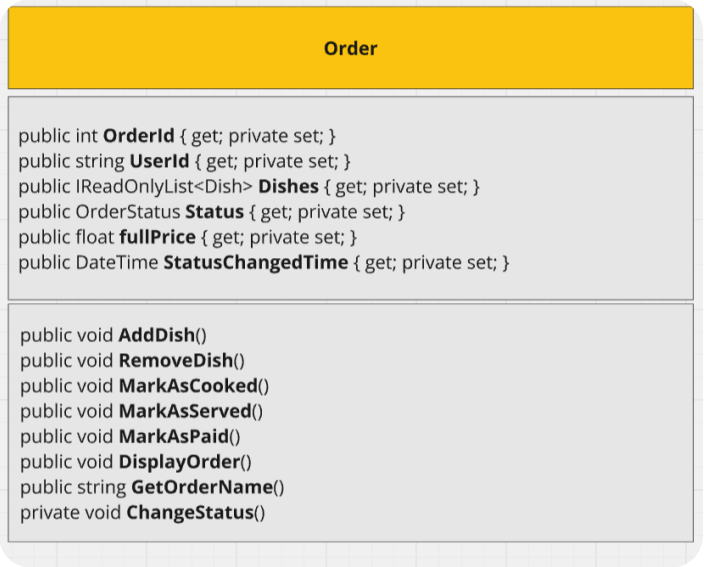
Category	Dish	Price (PLN)
1. 🥗 Appetizer		
	MozzarellaSticks	12,00 zł
	Guacamole	9,50 zł
	MozzarellaSticks	17,00 zł
	Nazwa	12,10 zł
2. 🍽️ Main		
	Spaghetti	6,00 zł
	Spaghetti	13,00 zł
	Pizza	17,90 zł
	Steak	14,00 zł
	Burger	12,00 zł
	Spaghetti	17,00 zł
	Burger	15,00 zł
	Lasagna	9,99 zł
	Pasta	21,22 zł
	Danie1	11,22 zł
	Pierogi	19,99 zł
	kaczka	40,00 zł
3. 🍰 Dessert		
	PannaCotta	11,10 zł
	PannaCotta	12,00 zł
4. 🥤 Beverage		
	Coffee	7,00 zł
	Lemonade	19,00 zł

Klasa Dish

reprezentuje pojedyncze danie w menu restauracji

Klasa Order

reprezentuje zamówienie w restauracji, umożliwiając dodawanie i usuwanie dań, zarządzanie statusem zamówienia (np. "Placed", "Cooked", "Served", "Paid") oraz wyświetlanie szczegółów zamówienia, w tym całkowitej ceny i listy dań.



Statusy zamówienia:

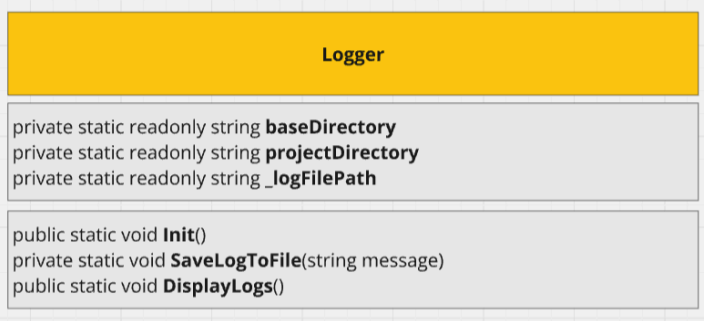
```
public enum OrderStatus {
    Placed,
    Cooked,
    Served,
    Paid
}
```

Wykorzystuje interface IOrder

```
Orders with status Placed:
┌──────────────┬──────────────┐
│ Dish         │ Price (PLN)  │
├──────────────┼──────────────┤
│ Guacamole    │ 9,50         │
│ Lasagna      │ 9,99         │
│ Pierogi      │ 19,99        │
├──────────────┼──────────────┤
│ Order ID:    │ 3            │
│ Status:      │ Placed       │
├──────────────┼──────────────┤
│ Total price: │ 39,48 PLN    │
└──────────────┴──────────────┘
```

Klasa Logger

Klasa Logger zapisuje logi do pliku (log.txt). Inicjalizuje zapis logów za pomocą metody Init(), zapisuje wiadomości do pliku w SaveLogToFile() oraz wyświetla zapisane logi w DisplayLogs().



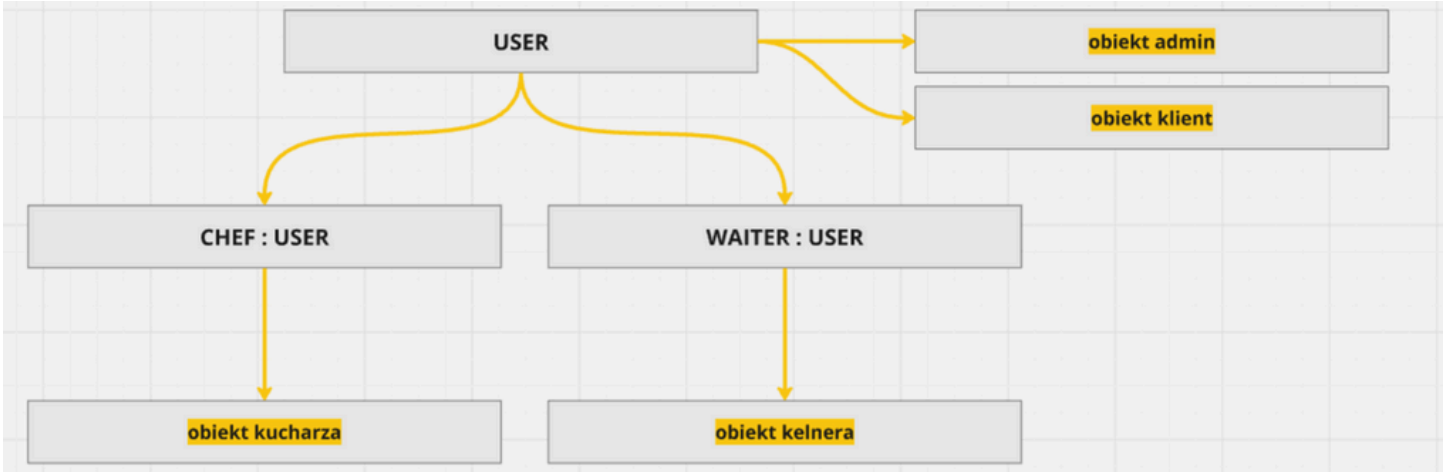
Zdarzenie:

Zdarzenie **OnLog** jest powiązane z metodą **SaveLogToFile**. Gdy zostanie wywołane zdarzenie, Logger zapisuje wiadomość do pliku logów (log.txt).

```
1 odwołanie
public static void Init()
{
    LogManager.OnLog += SaveLogToFile;
}
```

```
[2025-04-20 23:08:19] User 'customer' started creating a new order.
[2025-04-20 23:08:23] User 'customer' added dish 'kaczka' to the order.
[2025-04-20 23:08:27] User 'customer' added dish 'Coffee' to the order.
[2025-04-20 23:08:28] User 'customer' placed a new order.
[2025-04-20 23:08:40] User 'customer' logged out.
[2025-04-20 23:08:45] User 'waiter' logged in.
[2025-04-20 23:08:48] User 'waiter' logged out.
[2025-04-20 23:08:53] User 'chef' logged in.
[2025-04-20 23:09:53] User 'chef' logged in.
[2025-04-20 23:30:30] User 'customer' logged in.
[2025-04-20 23:30:33] User 'customer' started creating a new order.
[2025-04-20 23:30:36] User 'customer' added dish 'kaczka' to the order.
[2025-04-20 23:30:39] User 'customer' added dish 'Coffee' to the order.
[2025-04-20 23:30:40] User 'customer' placed a new order.
[2025-04-20 23:30:47] User 'customer' logged out.
[2025-04-20 23:30:55] User 'chef' logged in.
[2025-04-20 23:43:46] User 'customer' logged in.
[2025-04-20 23:43:49] User 'customer' started creating a new order.
[2025-04-20 23:43:52] User 'customer' added dish 'kaczka' to the order.
```

Klasa User



User jest klasą bazową

Chef i Waiter to klasy dziedziczące po klasie User

Użytkownicy wraz z danymi do logowania zapisywani są w pliku userPasswords.txt

```
a,ypeBEsobvcr6wjGzmiPcTaeG7/gUfE5yuYB3ha/uSLs=
waiter,ypeBEsobvcr6wjGzmiPcTaeG7/gUfE5yuYB3ha/uSLs=
chef,ypeBEsobvcr6wjGzmiPcTaeG7/gUfE5yuYB3ha/uSLs=
customer,ypeBEsobvcr6wjGzmiPcTaeG7/gUfE5yuYB3ha/uSLs=
customer2,ypeBEsobvcr6wjGzmiPcTaeG7/gUfE5yuYB3ha/uSLs=
customer3,ypeBEsobvcr6wjGzmiPcTaeG7/gUfE5yuYB3ha/uSLs=
angryCustomer,aIeH2P8UTFAsf1z/qv4sxYjYYHn53ogwTCawy5nOkcY=
politeCustomer,bztaBZoCryD2kkrr4I94jGY16j4a4DXcrpnX7f1Ebd4=
badCook,bztaBZoCryD2kkrr4I94jGY16j4a4DXcrpnX7f1Ebd4=
Gordon,RYOo+NlixKUk36iLgI/kL9gzx8IEmkrvOqJ7DttogG4=
```

Użytkownicy wraz z danymi do logowania zapisywani są w pliku userRoles.txt

```
customer3,Customer
a,Admin
waiter,Waiter
chef,Chef
a,Admin
customer,Customer
customer2,Customer
politeCustomer,Customer
Gordon,Chef
```

RBAC

Klasa RBAC posiada funkcję **HasPermission**, która sprawdza, czy użytkownik (user) posiada określone uprawnienie

Uprawnienie	Dostępność
ManageMenu	Tylko dla Admina
ViewMenu	Klient
ManageEmployees	Tylko dla Admina
ManageClients	Tylko dla Admina
DisplayOrders	Tylko dla Admina
CheckOrderStatus	Kucharz, Kelner
CheckMyOrders	Tylko dla Klienta
ChangeOrderStatus	Kucharz, Kelner
PlaceAnOrder	Tylko dla Klienta
PayForOrder	Tylko dla Klienta
ViewRevenue	Tylko dla Admina
ViewLogs	Tylko dla Admina

PasswordManager

Zarządza użytkownikami, ich hasłami i rolami. Umożliwia dodawanie użytkowników, weryfikację haseł, przypisywanie i usuwanie ról oraz wyświetlanie listy pracowników i klientów.

Metody:
VerifyPassword
HashPassword
AddUser
GetUserRoles
AddUserRole
RemoveUserRole
DisplayAllEmployees
DisplayCustomers

Właściwości:
_passwordFilePath
_rolesFilePath
PasswordVerified

W klasie PasswordManager hasła użytkowników są **hashowane za pomocą algorytmu SHA-256**. Proces ten znajduje się w prywatnej metodzie.

OrderManager

Klasa odpowiada za zarządzanie zamówieniami w systemie restauracyjnym.

Główne pola:

orders – lista wszystkich zamówień.

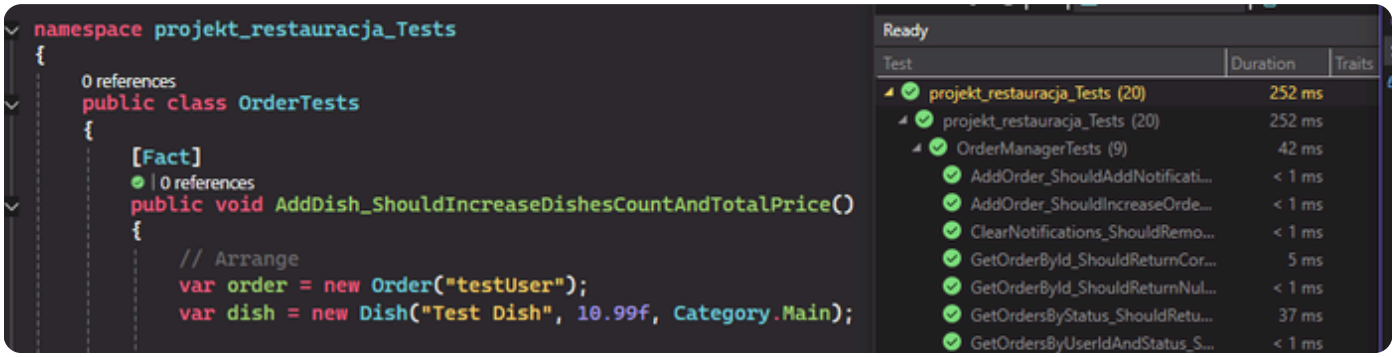
revenue – suma przychodów.

notificationsForChef, **notificationsForWaiter** – powiadomienia dla kucharza i kelnera.

- **AddOrder**(Order order)
- **GetOrders**()
- **DisplayOrderSummaryByStatus**()
- **HasOrdersWithStatus**(OrderStatus status)
- **GetOrderById**(int orderId)
- **DisplayOrdersByUserId**(string userId)
- **DisplayOrdersByStatus**(OrderStatus status)
- **GetOrdersByStatus**(OrderStatus status)
- **DisplayOrdersByUserIdAndStatus**(string userId, OrderStatus status)
- **GetOrdersByUserIdAndStatus**(string userId, OrderStatus status)
- **DisplayAllOrders**()
- **NotifyWaiter**(Order order)
- **GetNotificationsForRole**(UserRole role)
- **ClearNotifications**(UserRole role)

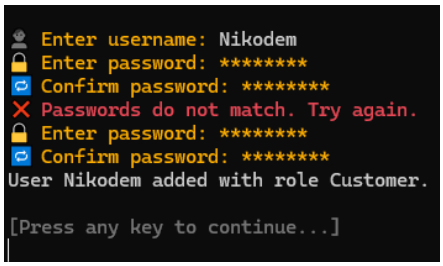
Dla klasy **OrderManager zostały wykonane testy jednostkowe** które:

- Sprawdzają, czy klasa OrderManager poprawnie zarządza zamówieniami i powiadomieniami.
- Testują m.in. dodawanie zamówień, filtrowanie ich po statusie i użytkowniku, powiadamianie kucharza i kelnera oraz obsługę sytuacji, gdy zamówienie nie istnieje.

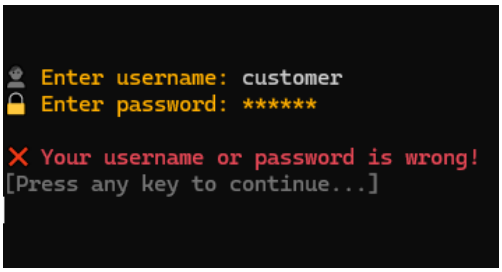


Obsługa błędów

- **Błędne logowanie i rejestracja**— komunikat o niepoprawnym haśle

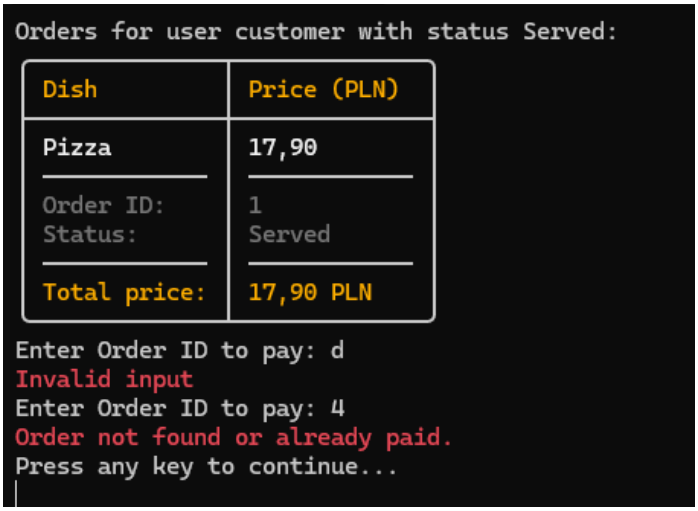


Rejestracja

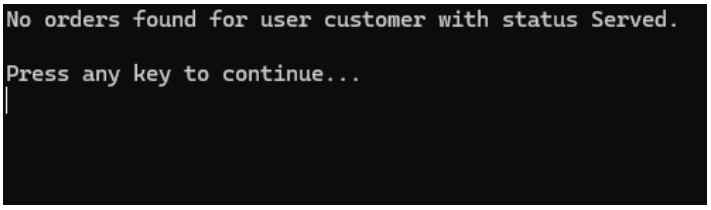


Logowanie

- **Niepoprawna zmiana statusu** — komunikat dla użytkownika

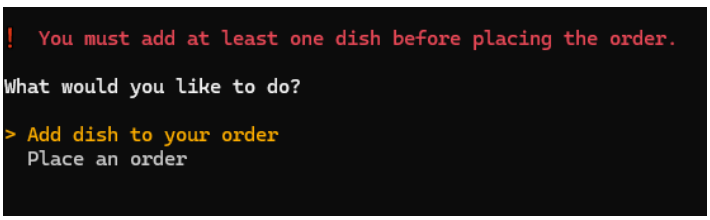


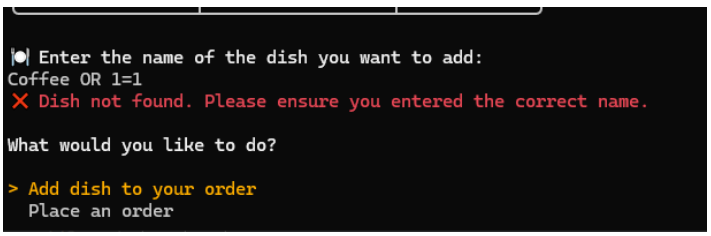
Niepoprawne id



Brak zamówień

- **Obsługa pustych list i braków danych**

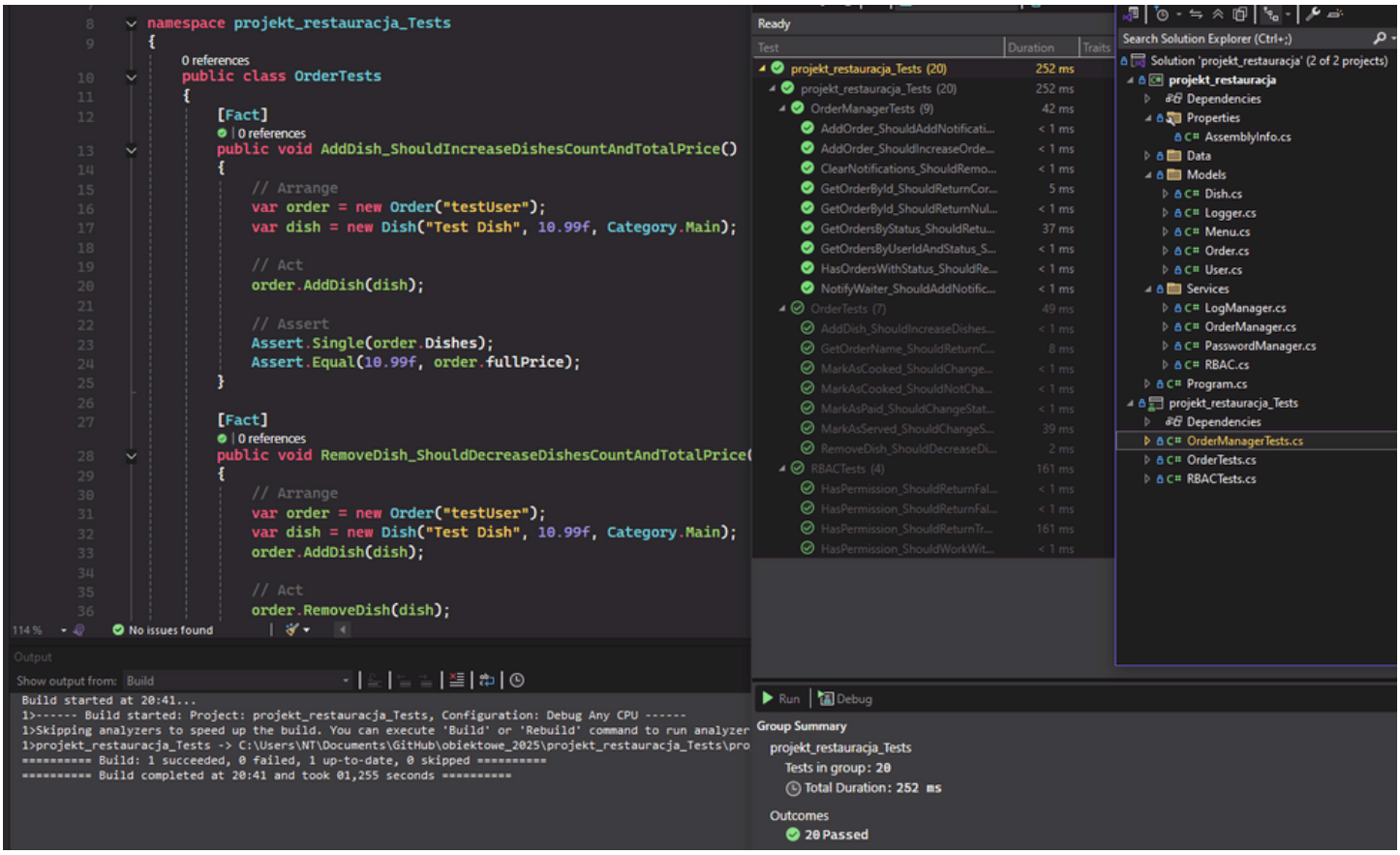




Testowanie

- **Testy ręczne** wszystkich funkcjonalności w konsoli
- **Testy jednostkowe** xUnit dla klas:
  - OrderManager
  - Order
  - RBAC

(Sprawdzanie danej funkcjonalności jest opisane nazwą testu jednostkowego)



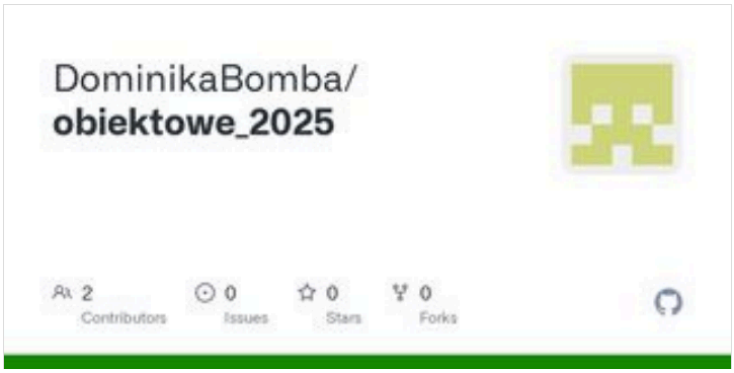
Problemy i ograniczenia

- Brak trwałej bazy danych (wszystko działa w RAM)
- Brak testów automatycznych
- Brak rozbudowanego zarządzania użytkownikami i uprawnieniami

Plany rozwoju

- Dodanie bazy danych (SQLite/MariaDB)
- Interfejs graficzny (np. z Avalonia lub MAUI)
- Dodanie rozbudowanego systemu rejestracji oraz możliwość resetowania haseł

Aplikacja konsolowa dostępna do pobrania:



Autorzy

- **Dominika Bomba**
  - Kontakt: dominika.bomba@uczen.zsk.poznan.pl
- **Nikodem Jokiel**
  - Kontakt: nikodem.jokiel@uczen.zsk.poznan.pl