



# 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

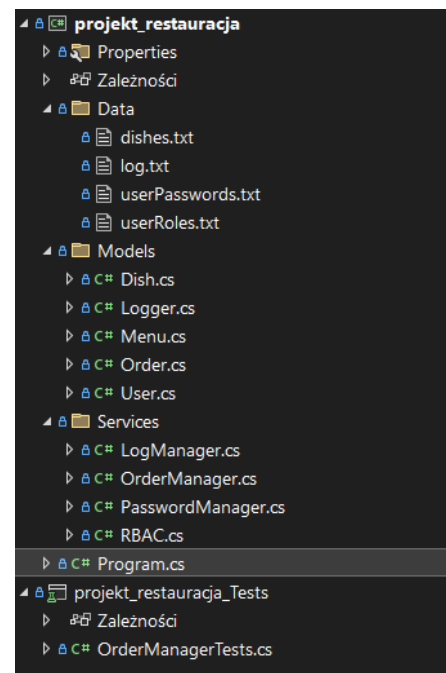
- **Język programowania:** C#
- **Środowisko:** .NET 8.0
- **IDE:** Visual Studio 2022
- **Biblioteki:**
  - Spectre.Console (do obsługi interfejsu w konsoli)
  - XUnit (Do testów jednostkowych)

# 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/obiektywne_2025`
- lub rozpakować plik .zip z moodle

## • Uruchomienie w konsoli:

- `cd [directory]\obiektywne_2025\projekt_restauracja\projekt_restauracja`
- `dotnet build`
- `dotnet run`

## • Uruchomienie w Visual Studio 2022:

▶ projekt\_restauracja ▶ ▶

# Opis działania aplikacji

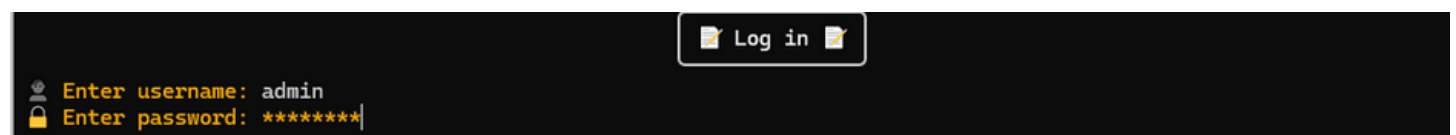
- Funkcjonalności:
  - Rejestracja i logowanie
  - Logowanie z rolami: admin, klient, kucharz, kelner
  - Składanie i obsługa zamówień
  - Zmiana statusu zamówienia (Placed > Cooked > Served > Paid)
  - Wyświetlanie zamówień z ostatnimi zmianami (dla admina)
  - Zarządzanie menu
- Przepływ programu:
  - 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

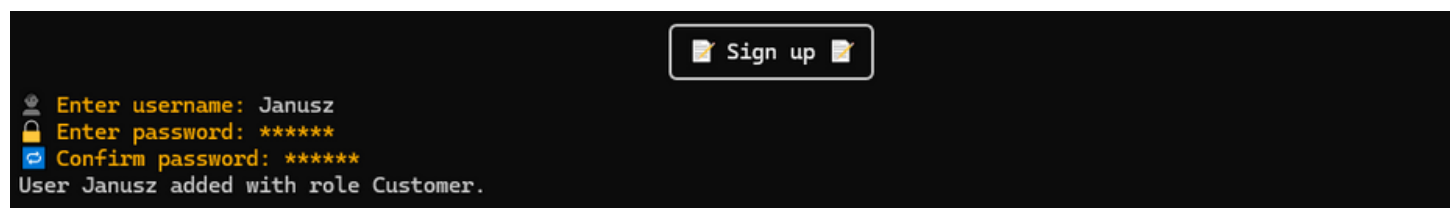


## Menu z wyborem:

- 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
	Daniel	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
<hr/>	
Order ID:	2
Status:	Placed
<hr/>	
Total price:	51,10 PLN

Złożenie zamówienia

```
Orders for user customer with status Served:

Dish      Price (PLN)
Guacamole 9,50
Lasagna   9,99
Pierogi    19,99

Order ID: 3
Status: Served

Total price: 39,48 PLN

Enter Order ID to pay: 3
The order has been paid.
Order paid!
Press any key to continue...
```

Kucharz gotuje zamówienie



zostaje podane przez kelnera



zostaje opłacone przez klienta



## Kelner

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

Select an option:

```
> Orders
  Log out
  Exit the program
```

```
Manage Orders

Option  Description
1      Check Order Status
2      Change Order Status
3      Back to main menu

Choose an option: |
```

## Funkcje

```
Logged in as: waiter

There are orders ready to serve!

-----Waiter Notifications-----
🔔 New Orders to Serve:
• OrderId: 1, Customer: customer [Coffee, kaczką, rosół]
• OrderId: 2, Customer: customer [Pizza]
• OrderId: 3, Customer: customer [Burger, rosół]
```

Powiadomienia o zamówieniach do podania

```
Orders with status Cooked:

Dish      Price (PLN)
Coffee     7,00
kaczka     40,00
rosół      12,00

Order ID: 1
Status: Cooked

Total price: 59,00 PLN

Press any key to continue...
```

Wyświetlanie zamówień gotowych do podania

```
Dish      Price (PLN)
Pizza     17,90

Order ID: 2
Status: Cooked

Total price: 17,90 PLN

Enter Order ID to change status: 2
The order has been served.
Order status updated!
Press any key to continue...
```

Serwowanie zamówień



# Kucharz

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

Select an option:

```
> Orders
  Log out
  Exit the program
```

Manage Orders

Option	Description
1	Check Order Status
2	Change Order Status
3	Back to main menu

Choose an option: |

## Funkcje

Logged in as: chef

There are orders to cook!

Chef Notifications

🔔 New Orders to Cook:

- OrderId: 1, Customer: customer [PannaCotta]
- OrderId: 2, Customer: customer [PannaCotta, kaczką]
- OrderId: 3, Customer: customer [Guacamole, Lasagna, Pierogi]
- OrderId: 4, Customer: customer [Coffee]

Powiadomienia o zamówieniach do ugotowania

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

Wyświetlanie złożonych zamówień

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

Enter Order ID to change status: 1=3  
Invalid input  
Enter Order ID to change status: 3  
The order has been cooked.  
Order status updated!

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

Select an option:

```
> Menu
  Orders
  Employees
  Clients
  Revenues
  Logs
  Log out
  Exit the program
```

# Przykładowe funkcje Administratora

Employee Management

Option	Description
1	Display all employees
2	Add a new employee
3	Fire an employee
4	Add a role to an employee
5	Remove a role from an employee
6	Back to main menu

Choose an option: 6  
Returning to main menu...

Press any key to continue...

zarządzanie pracownikami

Username	Roles
waiter	Waiter
chef	Chef
badCook	Chef

Enter the username of the employee to fire: badCook  
Employee badCook has been removed.

Press any key to continue...

zwalnianie pracowników

Enter new employee username: Gordon  
Enter new employee password: stolowkaZSK  
Select a role for the new employee:

> Chef  
Waiter

dodawanie pracowników

Username	Roles
waiter	Waiter
chef	Chef
Gordon	Chef

Enter the username to add a role to: Gordon  
Select a role to add:

> Admin  
Customer  
Waiter  
Chef

dodawanie roli użytkownikom

Orders by Status:

Cooked 1  
Paid 2

✓ Paid Orders:

Order ID	Customer ID	Price (PLN)
1	customer	21,22
2	customer	47,00
Total:		68,22 PLN

Other Orders:

Order ID	Customer ID	Price (PLN)	Status
3	customer	19,00	Cooked

przeglądanie przychodów oraz zamówień

[2025-04-21 18:49:54] User 'chef' logged in.  
[2025-04-21 18:49:59] User 'chef' cooked an order  
[2025-04-21 18:50:05] User 'chef' logged out.  
[2025-04-21 18:50:10] User 'waiter' logged in.  
[2025-04-21 18:50:13] User 'waiter' served an order  
[2025-04-21 18:50:20] User 'waiter' logged out.  
[2025-04-21 18:50:25] User 'customer' logged in.  
[2025-04-21 18:52:15] User 'customer' paid 17,90 zł for the order.  
[2025-04-21 18:59:16] User 'Janusz Kowalski' logged in.  
[2025-04-21 18:59:37] User 'Janusz Kowalski' logged out.  
[2025-04-21 19:17:53] User 'a' logged in.  
[2025-04-21 19:19:17] User 'a' logged in.  
[2025-04-21 19:20:46] User 'a' logged in.

[Press any key to return to menu...]

przeglądanie logów

Username	Roles
waiter	Waiter
chef	Chef
Gordon	Chef, Admin

Enter the username to remove a role from: Gordon  
Select a role to remove:

> Chef  
Admin

usuwanie roli użytkownikom

Username	Roles
customer3	Customer
customer	Customer
customer2	Customer
angryCustomer	Customer

Enter the username of the customer to remove: angryCustomer  
Customer angryCustomer has been removed.

Press any key to continue...

usuwanie klientów

Enter new customer's username: politeCustomer  
Enter new customer's password: zsk123  
User politeCustomer added with role Customer.

Press any key to continue...

dodawanie klientów

Select what to do:

> Cook Placed Orders  
Serve Cooked Orders

Orders with status Placed:

Dish	Price (PLN)
Burger	12,00
Order ID:	1
Status:	Placed
Total price:	12,00 PLN

Enter Order ID to update status: 1  
The order has been cooked.  
✓ Order status updated!

Press any key to continue...

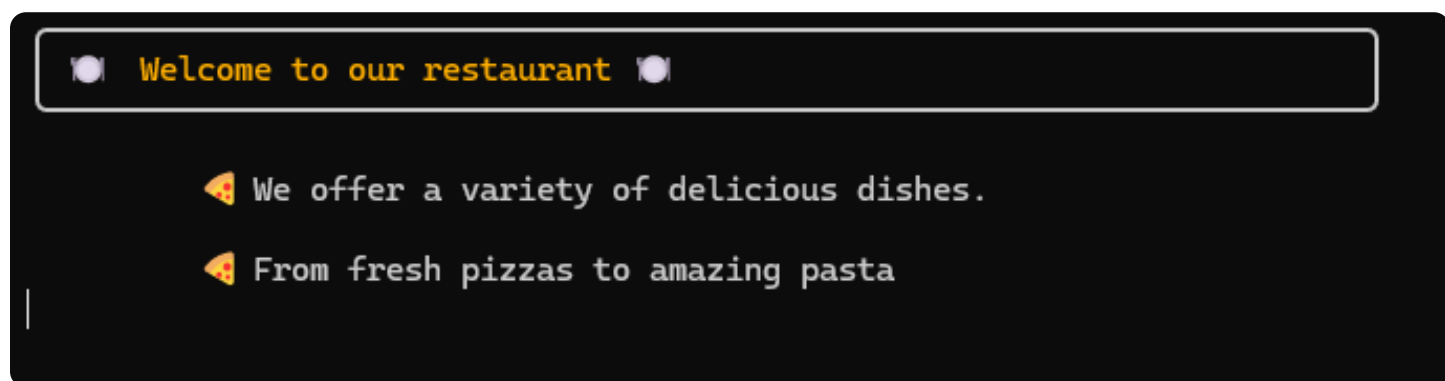
zmienianie statusu zamówienia

# Przykłady użycia

## Jak korzystać z aplikacji? – instrukcja krok po kroku

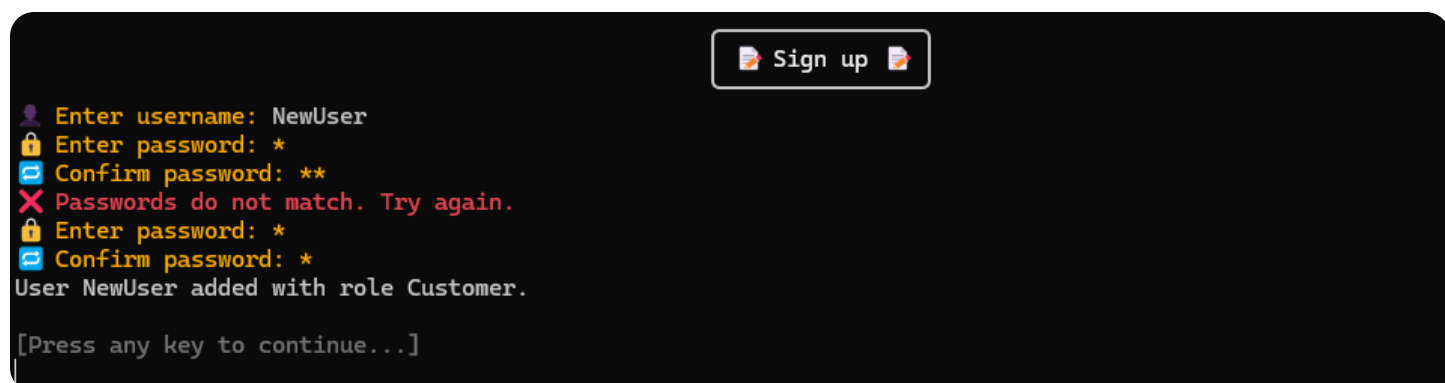
### 1. Uruchomienie aplikacji

- 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”.

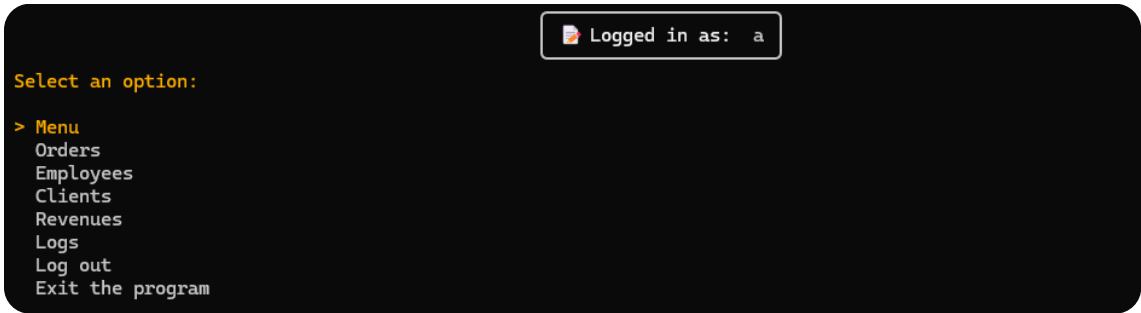
**Waiter** – kelner, zmienia statusy zamówień na „podane” i „zapłacone”.

**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
  - o 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 i może zmienić status podanego zamówienia na Zapłacone (**Paid**)

Logged in as: chef

There are orders to cook!

Chef Notifications

New Orders to Cook:

- OrderId: 1, Customer: customer [Coffee, kaczka, Pierogi]
- OrderId: 2, Customer: customer [Lasagna]

Select an option:

> Orders

Log out

Exit the program

## 6. Przegląd przychodów

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

Logged in as: a

Orders by Status:

Orders revenue

Paid  2

✓ Paid Orders:

Order ID	Customer ID	Price (PLN)
1	customer	66,99
2	customer	9,99
Total:		76,98 PLN

## 7. Logi operacji

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

```
[2025-04-22 15:03:52] User 'waiter' logged in.
[2025-04-22 15:04:04] User 'waiter' checked order status.
[2025-04-22 15:04:07] Waiter 'waiter' served an order
[2025-04-22 15:04:08] Waiter 'waiter' served an order
[2025-04-22 15:04:12] User 'waiter' logged out.
[2025-04-22 15:04:19] User 'customer' logged in.
[2025-04-22 15:04:23] User 'customer' paid 66,99 zł for their order (orderId: 1)
[2025-04-22 15:04:27] User 'customer' paid 9,99 zł for their order (orderId: 2)
[2025-04-22 15:04:32] User 'customer' logged out.
[2025-04-22 15:04:35] User 'a' logged in.
```

# 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.

Menu

```
public Dictionary<string, float> DishList { get; private set; }
public List<Category> Categories { get; private set; }
```

DishList  
lista wszystkich potraw pobierana z pliku

```
public LoadDishesFromFile(),
private SaveDishesToFile();
public GetDish();
public DisplayMenu()
public AddDish(),
public RemoveDish(),
public EditDish(),
public ModifyPrice();
```

Add, Remove, Edit, Modify  
Tylko dla Admina

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,Daniel
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
	Daniel	11,22 zł
3. 🍰 Dessert	Pierogi	19,99 zł
	kaczka	40,00 zł
	PannaCotta	11,10 zł
4. 🥤 Beverage	PannaCotta	12,00 zł
	Coffee	7,00 zł
	Lemonade	19,00 zł

## Klasa Dish

reprezentuje pojedyncze danie w menu restauracji

Dish

```
public string Name
public float Price
public Category Category
```

## 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

### Order

```
public int OrderId { get; private set; }  
public string UserId { get; private set; }  
public IReadOnlyList<Dish> Dishes { get; private set; }  
public OrderStatus Status { get; private set; }  
public float fullPrice { get; private set; }  
public DateTime StatusChangedTime { get; private set; }
```

```
public void AddDish()  
public void RemoveDish()  
public void MarkAsCooked()  
public void MarkAsServed()  
public void MarkAsPaid()  
public void DisplayOrder()  
public string GetOrderName()  
private void ChangeStatus()
```

Orders with status Placed:

Dish	Price (PLN)
Guacamole	9,50
Lasagna	9,99
Pierogi	19,99
<hr/>	
Order ID:	3
Status:	Placed
<hr/>	
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;  
}
```

### Logger

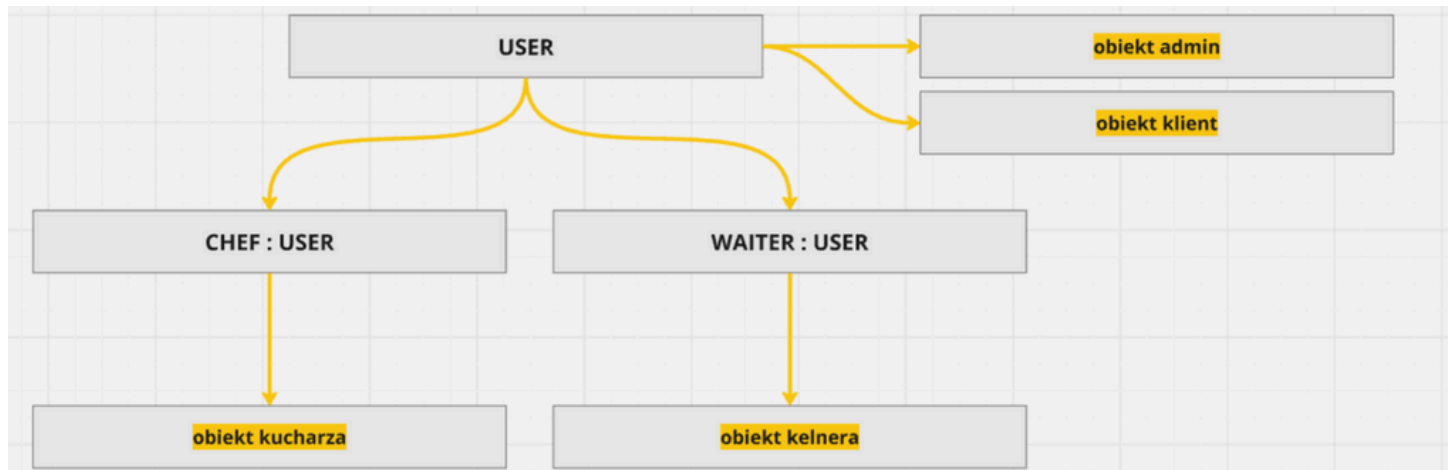
```
private static readonly string baseDirectory  
private static readonly string projectDirectory  
private static readonly string _logFilePath
```

```
public static void Init()  
private static void SaveLogToFile(string message)  
public static void DisplayLogs()
```

```
[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      Użytkownicy wraz z danymi do logowania zapisywani są w pliku userRoles.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/qv4sYjYYHn53ogwTCawy5n0kcY=
politeCustomer,bztaBZoCryD2kkrr4I94jGY16j4a4DXcrpnX7f1Ebd4=
badCook,bztaBZoCryD2kkrr4I94jGY16j4a4DXcrpnX7f1Ebd4=
Gordon,Ry0o+NlixKUK36iLgI/kL9gzx8IEmkrv0qJ7DttoG4=
```

```
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 hasel, 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 użytkownika, powiadamianie kucharza i kelnera oraz obsługę sytuacji, gdy zamówienie nie istnieje.

```
namespace projekt_restauracja_Tests
{
    0 references
    public class OrderTests
    {
        [Fact]
        0 references
        public void AddDish_ShouldIncreaseDishesCountAndTotalPrice()
        {
            // Arrange
            var order = new Order("testUser");
            var dish = new Dish("Test Dish", 10.99f, Category.Main);
        }
    }
}
```

Test	Duration	Traits
projekt_restauracja_Tests (20)	252 ms	
projekt_restauracja_Tests (20)	252 ms	
OrderManagerTests (9)	42 ms	
AddOrder_ShouldAddNotificati...	< 1 ms	
AddOrder_ShouldIncreaseOrde...	< 1 ms	
ClearNotifications_ShouldRemo...	< 1 ms	
GetOrderById_ShouldReturnCor...	5 ms	
GetOrderById_ShouldReturnNul...	< 1 ms	
GetOrdersByStatus_ShouldRetu...	37 ms	
GetOrdersByUserIdAndStatus_S...	< 1 ms	





# Obsługa błędów

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

```
Enter username: Nikodem
Enter password: *****
Confirm password: *****
X Passwords do not match. Try again.
Enter password: *****
Confirm password: *****
User Nikodem added with role Customer.

[Press any key to continue...]
```

Rejestracja

```
Enter username: customer
Enter password: *****

X Your username or password is wrong!
[Press any key to continue...]
```

Logowanie

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

```
Orders for user customer with status Served:
```

Dish	Price (PLN)
Pizza	17,90
Order ID:	1
Status:	Served
Total price:	17,90 PLN

```
Enter Order ID to pay: d
Invalid input
Enter Order ID to pay: 4
Order not found or already paid.
Press any key to continue...
```

Niepoprawne id

```
No orders found for user customer with status Served.
Press any key to continue...
```

Brak zamówień

- **Obsługa pustych list i braków danych**

```
! You must add at least one dish before placing the order.

What would you like to do?

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

```
Enter the name of the dish you want to add:
Coffee OR 1=1
X Dish not found. Please ensure you entered the correct name.

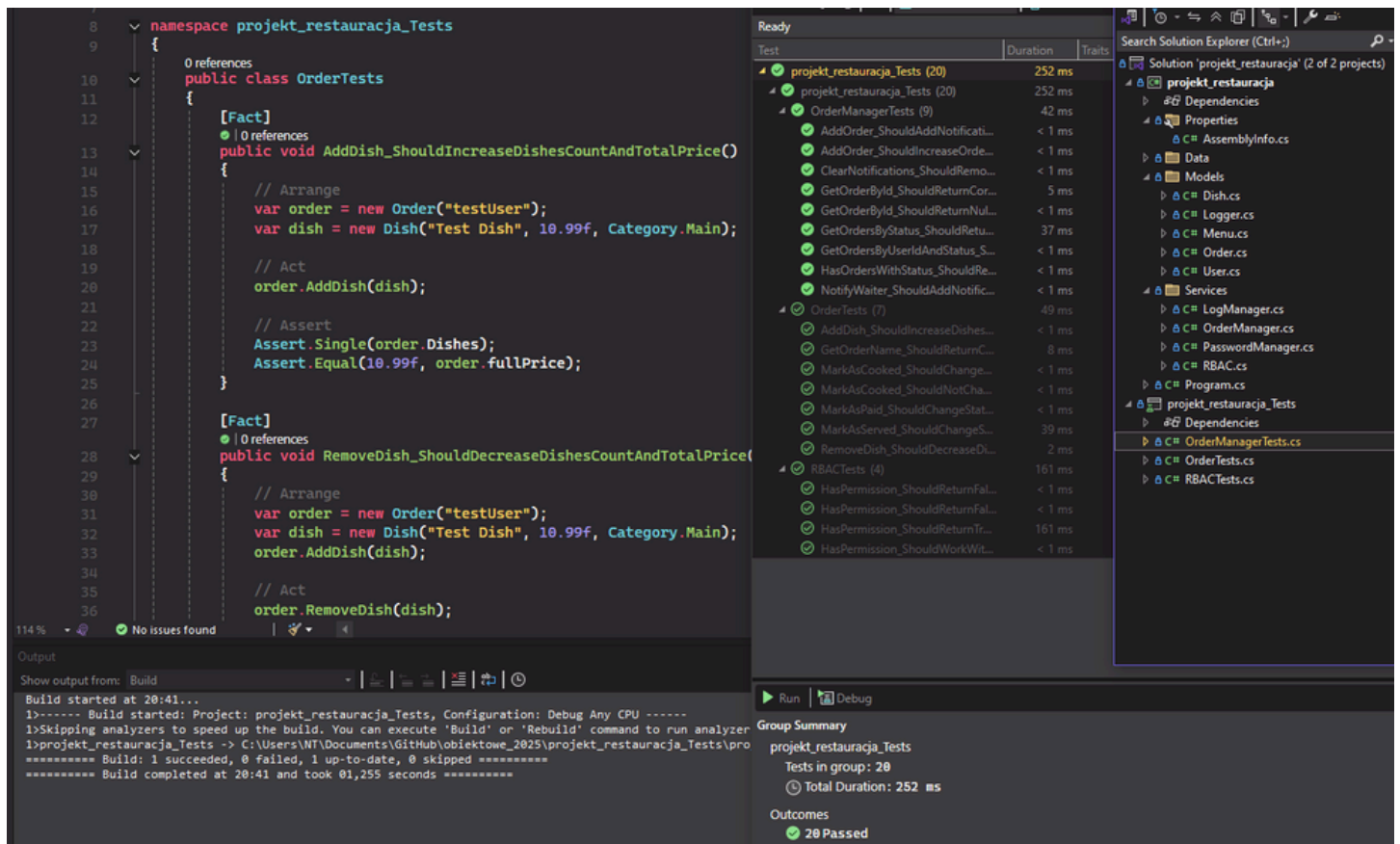
What would you like to do?

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

# 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

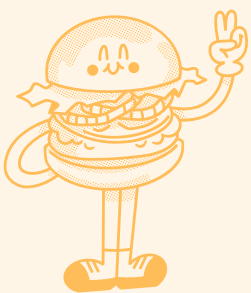
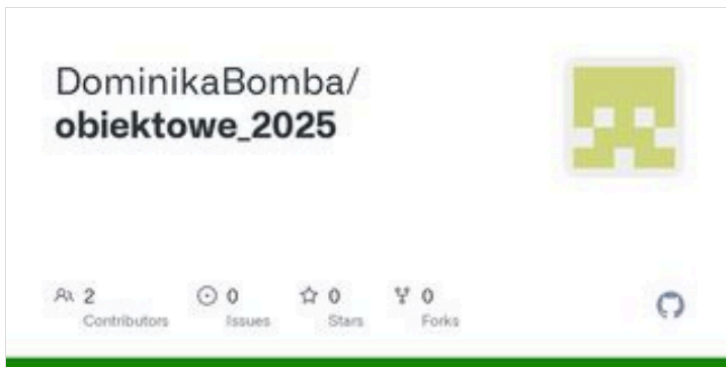
- Brak trwalej bazy danych (wszystko działa w RAM)
- Brak testów automatycznych
- Brak rozbudowanego zarządzania użytkownikami i uprawnieniami

# 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