WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: Programowanie

Paweł Skrok
Nr albumu w67266

Chłodnia owoców C#

Prowadzący: mgr inż. Ewa Żesławska

Praca projektowa programowanie obiektowe C#

Rzeszów 2024

Spis treści

Rozdział 1

Zamysł Projektu

System zarządzania chłodnią to aplikacja konsolowa opracowana w celu uproszczenia procesu zarządzania owocami oraz pracownikami w chłodni.

Celem Programu "Chłodnia owoców C#"jest dostarczenie narzędzia, które ułatwi zarządzanie bazą chłodni owoców. **Główne cele to :**

- Stworzenie funkcji umożliwiającej dodawanie, aktualizację i usuwanie danych dotyczących przechowywanych owoców.
- Stworzenie odpowiedniego systemu zarządzania który jest wygodny i prosty w obsłudze.
- Wyświetlanie szczegółów dotyczących przechowywanych owoców i zatrudnionych pracowników w chłodni.
- Umożliwia dostęp do systemu poprzez autoryzację, np. login i hasło

Rozdział 2

Wymagania funkcjonalne i niefunkcjonalne

2.1 Wymagania fukcjonalne

- podgląd bazy owoców i pracowników
- Możliwość dodawania Pracowników
- Aplikacja sprawdza czy pracownik jest zatrudniony
- Dodawanie nowych owoców do zapasów, aktualizację danych o istniejących oraz usuwanie ich

2.2 Wymagania niefunkcjonalne:

- Aplikacja powinna być intuicyjna i łatwa w obsłudze
- Czas reakcji systemu na działanie użytkownika powinien być minimalny
- Aplikaja powinna być zapisana zgodnie z zasami programowania obiektowego w języku c#

Rozdział 3

Opis struktury projektu

3.1 Opis techniczny

Język programowania:

- Projekt został napisany w języku C#.

Narzędzia:

- Wykorzystano platformę .NET do budowy i uruchamiania aplikacji.
- Do manipulacji danymi JSON użyto biblioteki JSON.NET.

Zarządzanie danymi:

- W Programie zdefiniowano klasy: Fruit, Employee,ToDoWithFruits,Add_Employees,DisplayData oraz jedna główna ToDoFruitColdStorage która spina pozostałe zgodnie z hierarchią dziedziczenia

Dane:

- Pobieranie, wczytywanie i zapis odbywa się na 2 plikach employeeData.json oraz Fruits_base.json w lokalizacji C:\Users\Acer\source\Project11\

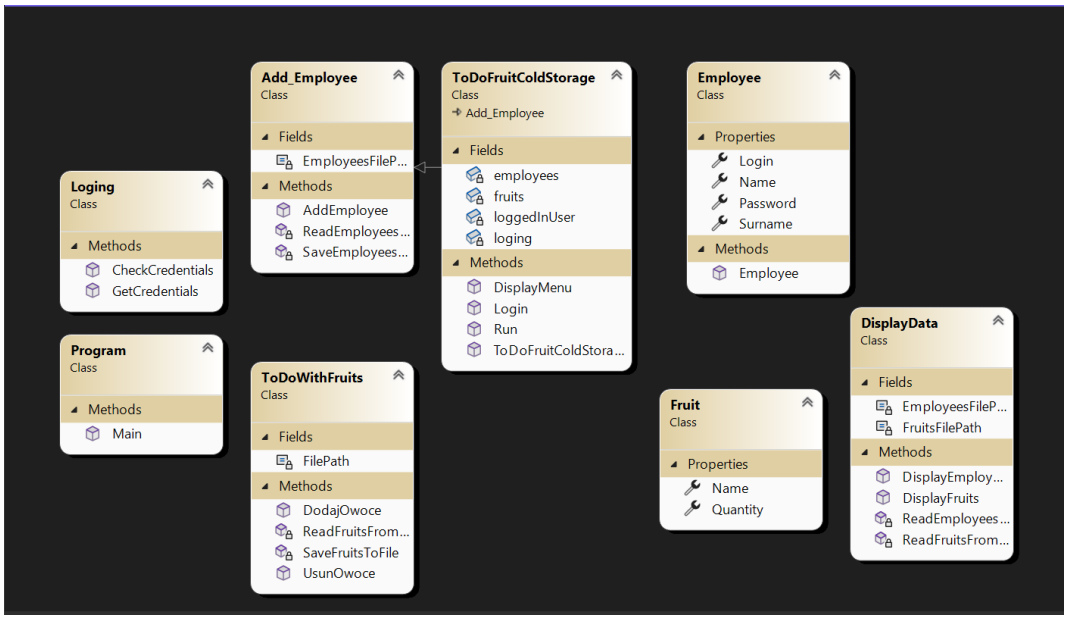
Aplikacja korzysta z plików .json z biblioteki:Newtonsoft.Json
Wymagania sprzętowe dla Aplikacji:

- System operacyjny zgodny z z Microsoft Visual studio oraz platformą .NET
- minimum 4 GB pamięci RAM,

3.2 Diagram klas

Diagram przedstawia strukturę klas w projekcie oraz ich relacje dziedziczenia i kompozycji. Kazda klasa reprezentuje określone pola lub metody

- **Klasa Fruit** zawiera Pola nazwa owocu i waga
- **Klasa Employee** zawiera Pola oraz konstruktor które wykorzystuje klasa Add_Employee do utworzenia obiektu.
- **Klasa ToDOWithFruit** zawira pole ścierzki do pliku zapisu i 4 główne metody programu
- **Klasa DisplayData** zawiera ścierzki do plików i ma za zadanie wczytanie danych z plików przechowujących dane o owocach jak i pracownikach i wypisanie ich.
- **Klasa Add_Employee** zawiera ścierzkę dostępu do Pliku zapisu pracowników w formacie .json , oraz metody tworzenia obiektu pracownika oraz zapisu do pliku.
- **Klasa Logging** jest w trakcie pracy zawiera ona dane logowania osoby która jest nad pracownikami która miała by dostęp do dodawania pracowników jak i usuwania, w przyszłości. W tym momencie sprawdza ona warunek czy instnieje dany pracownik.
- **Klasa ToDoFruitColdStorage** najważniejszą metodą jest DisplayMenu które wyświetla menu wyboru, Run która jest główną pętlą programu i Login która jest przygotowana pod sprawdzenie czy istnieje pracownik
- **Klasa Program** przyjmuje jako pierwszy stworzony obiekt Loging który pyta o dane logowania, oraz ToDoFruitColdStorage odpowiedzialny za działanie całego programu



Rysunek 3.1: Diagram Klas

3.3 Opis klas

Klasa ToDoFruitColdStorage reprezentuje główną strukture w systemie ma kilka kluczowych pól oraz właściwości:

- pola są typu: private List<Object> objects;
- metoda DisplayMenu które wyświetla menu wyboru
- metoda Run jes główną pętlą programu
- metoda Login to zalogowania się do programu (metoda jest nie dokończona)
- Konstruktor który tworzy obiekty owoce, pracownicy, logowanie(login hasło)

Klasa Add_Employee Jest klasą główną zarządzającą danymi pracownika :

- przechowuje ścierzkę do pliku zapisu pracowników
- Metoda AddEmplyee odpowiada za dodanie pracownika
- metoda ReadEmployees odczytuje z pliku zapisu pracowników
- metoda SaveEmplyeesToFile zapisuje utworzonego pracownika do pliku employeeData.json

Klasa DisplayData obsługuje odczyt z pliów i wyświetlenie dostępnych danych na ekranie konsoli

- metoda ReadEmployeeFromFile odczytuje dane zapisane w pliku pracownicy jako lista słowników
- metoda ReadFruitFromFile odczytuje owoce zapisane w pliku
- Metoda DisplayEmployees wyśietla wcześniej odczytane dane pracowników
- Metoda DisplayFruits wyświetla wcześniej odczytane dane owoców

Klasa ToDOWithFruits jest klasą zarządzającą owocami pożycza pola po klasie Fruit i przechowuje ścierzkę do pliku zapisu owoców

- Metoda DodajOwoce dodaje owoce do bazy danych lub jeśli już istnieją zwiększa wagę
- Metoda UsunOwoce usuwa podaną ilość z pliku jeśli wartość jest większa od ilości dostępnej owoc jest całkowicie usuwany

Klasa Emplyee zawiera pola które są przyczane przez inne klasy do tworzenia list obiektów

- pola klasy to: Login, Name, Password, Surname

Klasa Fruit zawiera pola które są przyczane przez inne klasy do tworzenia list obiektów

- pola klasy to: Name, Quantity

Klasa Loging pytan o login, hasło i sprawdza czy pracownik jest w bazie

- Metoda GetCredentials pyta o dane logowania
- Metoda CheckCredentials sprawdza czy wprowadzone dane istnieją w bazie (czy jest pracownik o podanym haśle i loginie)

Klasa Fruit zawiera pola które są przyczane przez inne klasy do tworzenia list obiektów

- pola klasy to: Name, Quantity

3.4 Baza danych

Baza danych opiera sie na plikach .json format przechowywanych danych jest listą w której obiekty (pracownik,owoc) są zapisywane jako słownik, np:[{ "Name": "Pawel", "Surname": "Skrro", "Login": "login12", "Password": "haslo123"}] ,[{ "Name": "Malina", "Quantity": 82.0 },

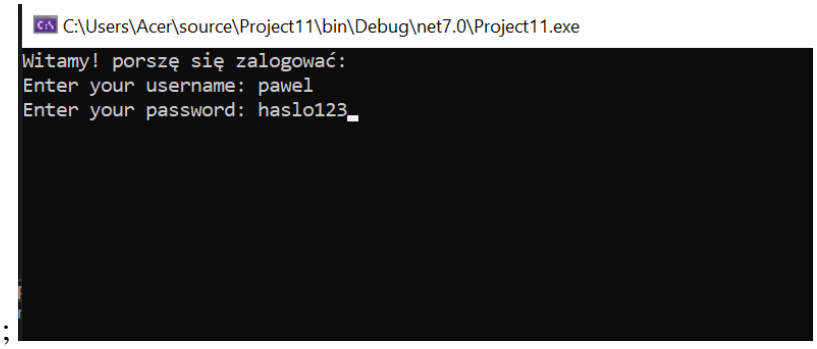
Plik zapisu pracowników employeeData.json

Plik zapisu Owoców Fruits_base.json

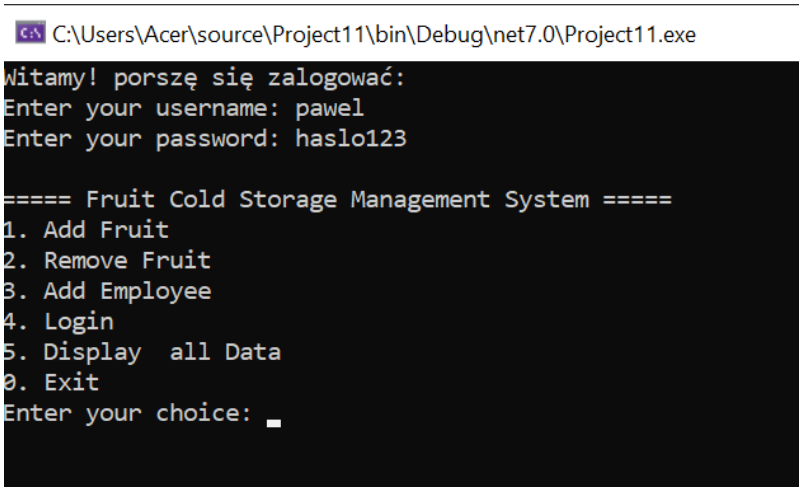
Rozdział 4

Prezentacja warstwy użytkowej

Przy otworzeniu aplikacji mamy powitanie oraz funckję która kiedyś będzie rozwijana czyli pierwsze logowanie do aplikacji, Następnie pokazuje sie Menu Wyboru które jest zapętlone aż użytkownik wciśnie na klawiaturze 0.

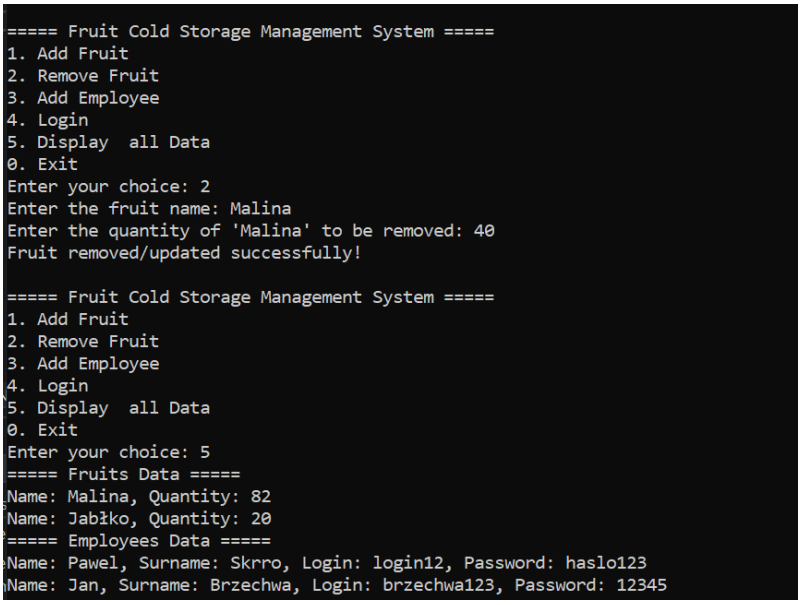


Rysunek 4.1: Opcja dodawania owocu



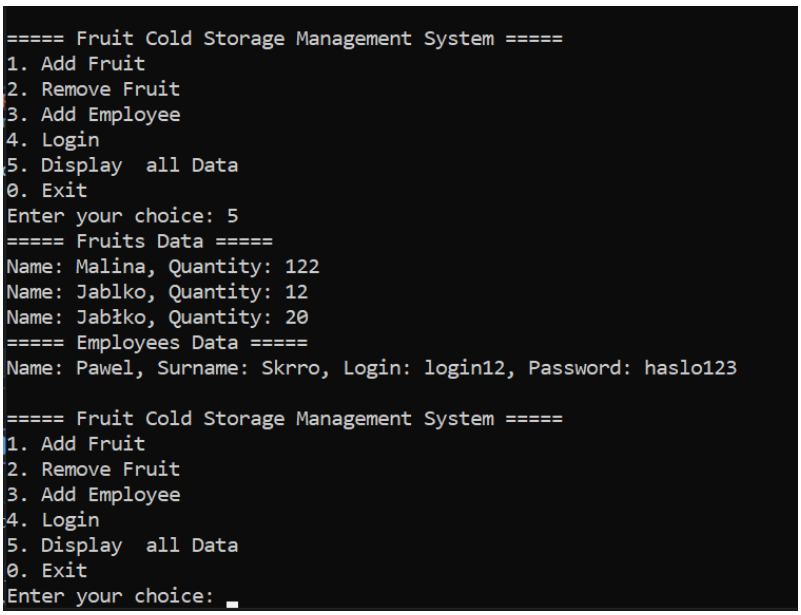
Rysunek 4.2: Menu Wyboru

Usuwanie owcu z bazy danych jeśli takowy się znajduje, jeśli ilość do usunięcia jest większa od ilości dostępnej owoc jest usuwany, ;



Rysunek 4.3: Usuwanie owocu

Poniżsszy wybór opcji 5 pobiera dane z plików i je wypisuje w razie braku nie zwraca nic ;



Rysunek 4.4: Wypisanie danych Pracowników oraz owoców

Jeśli chcemy dodać pracownika wypełniamy jego dane które zostaną zapisane do pliku

```
===== Fruit Cold Storage Management System =====
1. Add Fruit
2. Remove Fruit
3. Add Employee
4. Login
5. Display  all Data
0. Exit
Enter your choice: 3
Enter employee name: Jan
Enter employee surname: Brzechwa
Enter employee login: brzechwa123
Enter employee password: 12345
Employee 'brzechwa123' added successfully.
```

Rysunek 4.5: Dodanie Pracownika

```
Enter your choice: 5
===== Fruits Data =====
Name: Malina, Quantity: 122
Name: Jabłko, Quantity: 20
===== Employees Data =====
Name: Pawel, Surname: Skrro, Login: login12, Password: haslo123
Name: Jan, Surname: Brzechwa, Login: brzechwa123, Password: 12345
```

Rysunek 4.6: Aktualizacja po dodaniu pracownika

4 to zapytanie o dane pracownika. w ten sposób mozemy sprawdzić czy pracownik jest w bazie danych

```
===== Fruit Cold Storage Management System =====
1. Add Fruit
2. Remove Fruit
3. Add Employee
4. Login
5. Display  all Data
0. Exit
Enter your choice: 4
Enter your username: login12
Enter your password: haslo123
Login successful!

===== Fruit Cold Storage Management System =====
1. Add Fruit
2. Remove Fruit
3. Add Employee
4. Login
5. Display  all Data
0. Exit
Enter your choice:
```

Rysunek 4.7: Sprawdzenie danych pracownika

zamknięcie programu za pomocą opcji 0 ;

```
===== Fruit Cold Storage Management System =====
1. Add Fruit
2. Remove Fruit
3. Add Employee
4. Login
5. Display  all Data
0. Exit
Enter your choice: 0
Exiting the program. Goodbye!

C:\Users\Acer\source\Project11\bin\Debug\net7.0\Project11.exe (process 15512) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Rysunek 4.8: Wyjście z programu

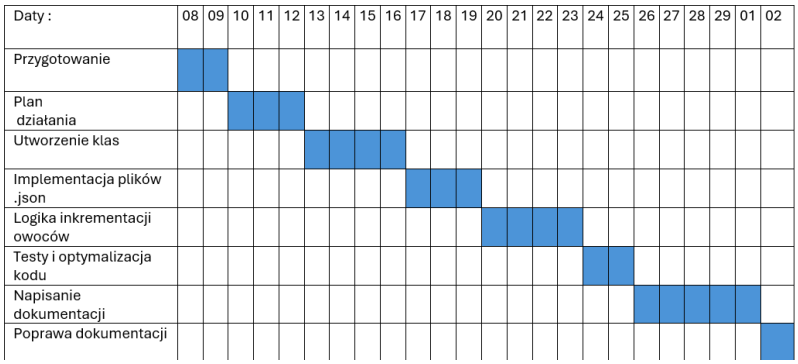
Rozdział 5

Harmonogram prac nad projektem

Prace nad projektem trwały od 08.02.2024-02.03.2024. Praca została podzielona na następujące etapy:

- Przygotowanie srodowiska VS Code - w tym etapie zainstalowanie odpowiednich bibliotek np. Newtonsoft.Json oraz utworzono repozytorium, które znajduje sie pod adresem https://github.com/zdrapek-jpg/c-wsiz-Application-FruitColdStorage
- Planowanie działania aplikacji - przygotowanie planu , wymyślenie logiki programu, Hierarchi klas oraz głównch celów apliakcji
- Utworzenie klas programu.
- Zaimplementowanie plików .json do zapisu danych oraz zapis logiki z obsługą wyjątków dla danych
- Utworzenie logiki inkremetnacji wagi owoców jeśli nazwa jest taka sama
- Testowanie - poprawienie działania programu oraz eliminacja błędów i optymalizacja kodu
- Napisanie dokumentacji oraz wypchnięcie kodu do repozytorium na githubie
- poprawa dokumentacji

Diagram Gantta



Rysunek 5.1: Diagram Gantta

Rozdział 6

Repozytorium

https://github.com/zdrapek-jpg/c-wsiz-Application-FruitColdStorage.git

Rozdział 7

Podsumowanie

Projekt "Chłodnia owoców"stanowi kompleksową aplikację konsolową do ułatwienia pracy. W ramach projektu zaimplementowano hierarchię dziedziczenia oraz powiązanie ze sobą 5 klas. Każda klasa posiada unikalne właściwości i ograniczenia, co dodatkowo wzbogaca funkcjonalność systemu.

Podczas rozwoju projektu zwrócono uwagę na wymagania funkcjonalne i niefunkcjonalne, co zaowocowało aplikacją, która nie jest tylko efektywna, ale także łatwa w obsłudze i responsywna. Harmonogram realizacji projektu został przestrzegany, a kod źródłowy został udostępniony na platformie GitHub, umożliwiając dalsze rozszerzenia i rozwijanie projektu.

7.1 Dalsze prace

- Utworzenie interfejsu graficznego
- Dodanie Logowania użytkownika do programu
- dodanie pliku który zapisuje historię działań użytkownika
- dodatkowe inforamcje o owocach jak i pracownikach
- dodanie nr pesel do pracownika i możliwości usuwania pracowników

Bibliografia

[1] https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/deserialization

[2] https://code-maze.com/csharp-read-and-process-json-file/

[3] https://www.youtube.com/watch?v=GhQdlIFylQ8t=4801s

[4] https://learn.microsoft.com/pl-pl/dotnet/csharp/fundamentals/exceptions/exception-handling

Spis rysunków