

Programowanie Obiektowe i Graficzne

dokumentacja projektu Plantcare

Bartosz Wanot

Jakub Maćkowiak

Sandra Świeczak

Informatyka, grupa 3F

Wydział Matematyki Stosowanej

15 września 2021

Spis treści

1	Aplikacja	3
1.1	Opis i problematyka programu	3
1.2	Instrukcja obsługi	3
1.2.1	Uruchomienie i ekrany startowe	3
1.2.2	Obsługa pielęgnacji	6
2	Baza danych	10
2.1	Diagram związków encji	12
2.2	Model relacyjny	13
2.3	Procedury	13
2.4	Algorytm planowanego podlewania	14
3	Implementacja	15
4	Możliwości rozwoju projektu	16

1 Aplikacja

1.1 Opis i problematyka programu

Aplikacja Plantcare została stworzona z myślą o zarządzaniu opieką nad roślinami doniczkowymi. Użytkownik tworzy konto oraz podaje rośliny, które ma pod swoją opieką. Aplikacja pozwala mu na kontrolę ich pielęgnacji w jednym miejscu.

Aplikacja jest aplikacją desktopową, opartą na języku C według koncepcji wzorca projektowego MVVM (Wzorzec Model-View-ViewModel).

Całość kodu aplikacji dostępna jest pod linkiem: <https://github.com/JakubMckowiak/CarePlant>

1.2 Instrukcja obsługi

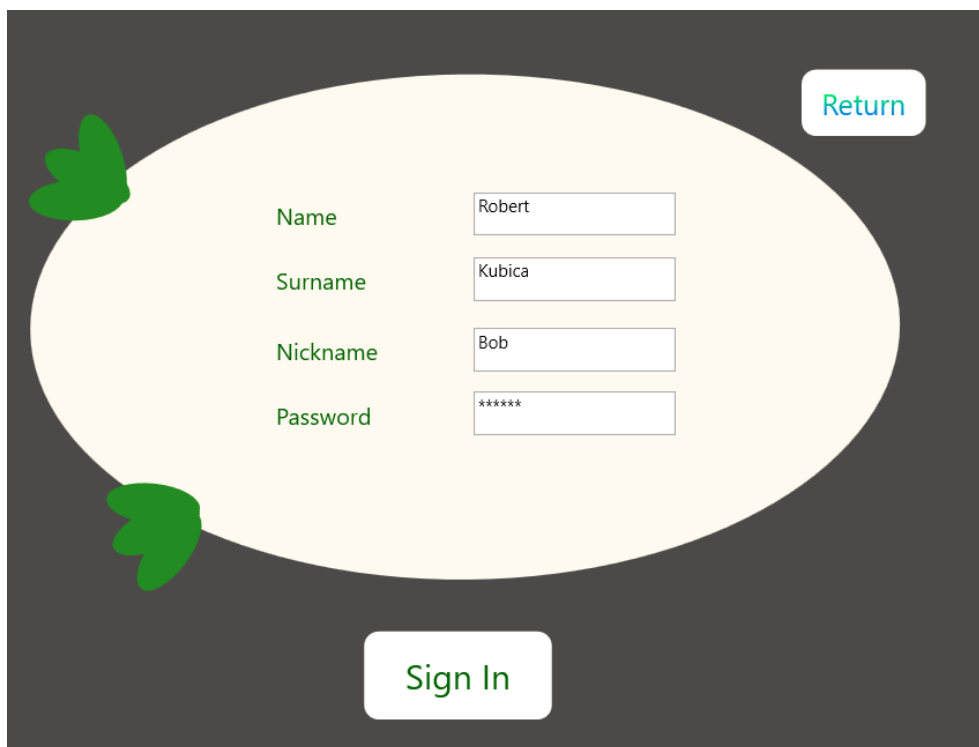
1.2.1 Uruchomienie i ekrany startowe

Program jest zabezpieczony przed niepożądanymi zmianami w bazie. Użytkownik dostaje komunikaty zwrotne podczas prób niedozwolonego logowania oraz dodawania pustych pól.

Użytkownik po uruchomieniu programu witany jest ekranem powitalnym, w którym wybiera rejestrację nowego użytkownika lub zalogowanie już istniejącego w bazie.



Rysunek 1: Ekran powitalny



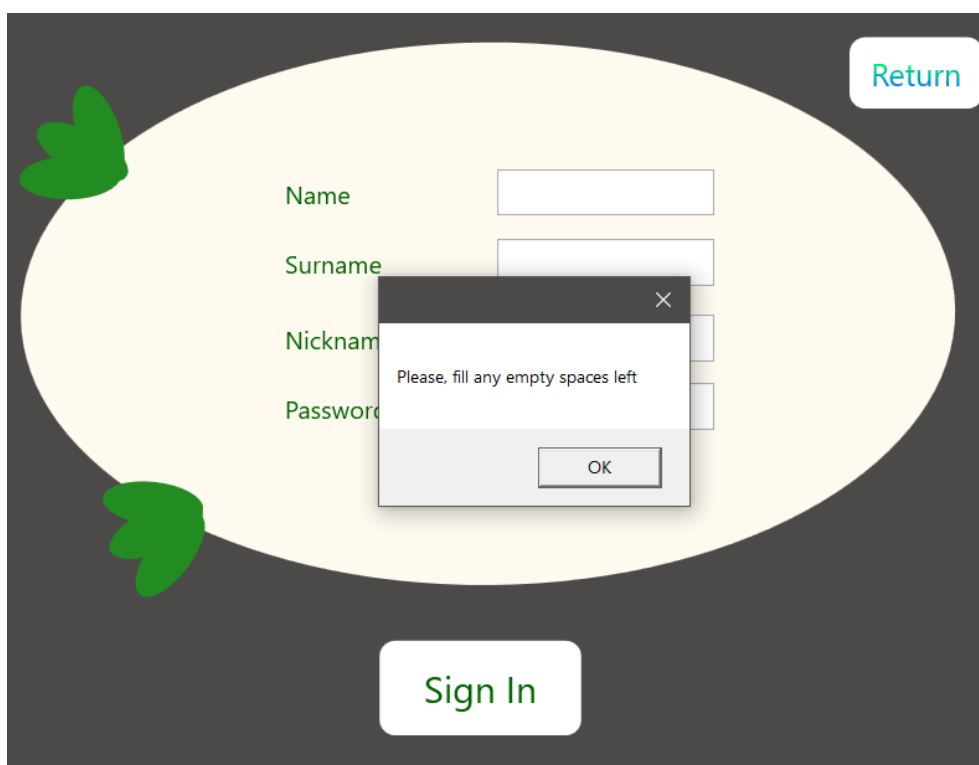
A registration form on a dark grey background. The form is contained within a light yellow oval with green leaf-like decorations on the left. It has four input fields: 'Name' with 'Robert', 'Surname' with 'Kubica', 'Nickname' with 'Bob', and 'Password' with '*****'. A 'Return' button is in the top right, and a 'Sign In' button is at the bottom center.

Name	Robert
Surname	Kubica
Nickname	Bob
Password	*****

Return

Sign In

Rysunek 2: Rejestracja nowego użytkownika



The same registration form as in Figure 2, but with an error dialog box overlaid. The dialog box has a close button (X) in the top right, the text 'Please, fill any empty spaces left', and an 'OK' button at the bottom. The input fields for Name, Surname, and Nickname are empty, while the Password field still contains '*****'.

Name	
Surname	
Nickname	
Password	*****

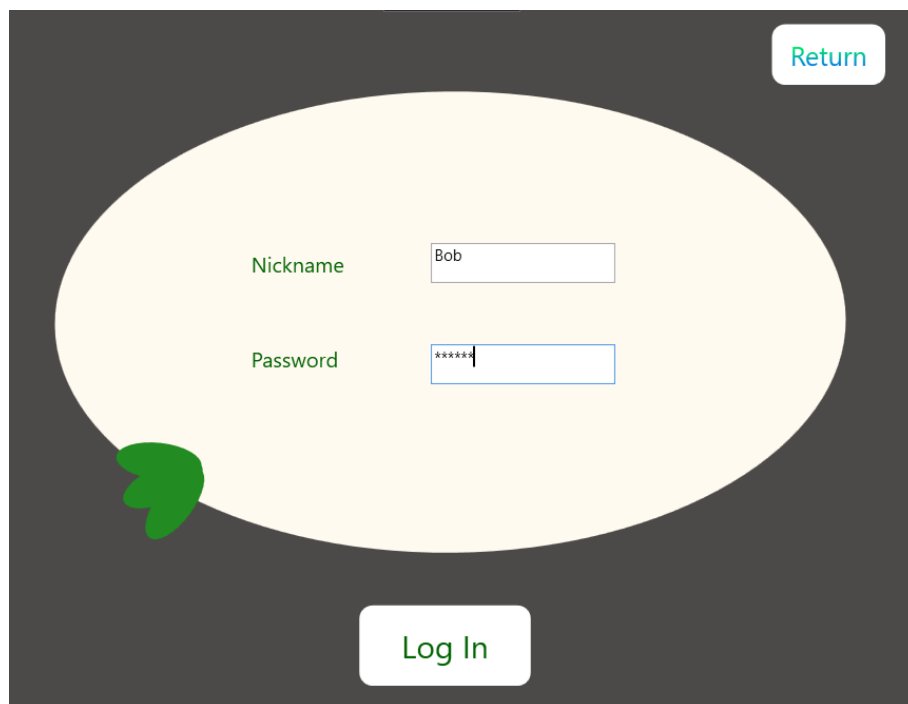
Return

Sign In

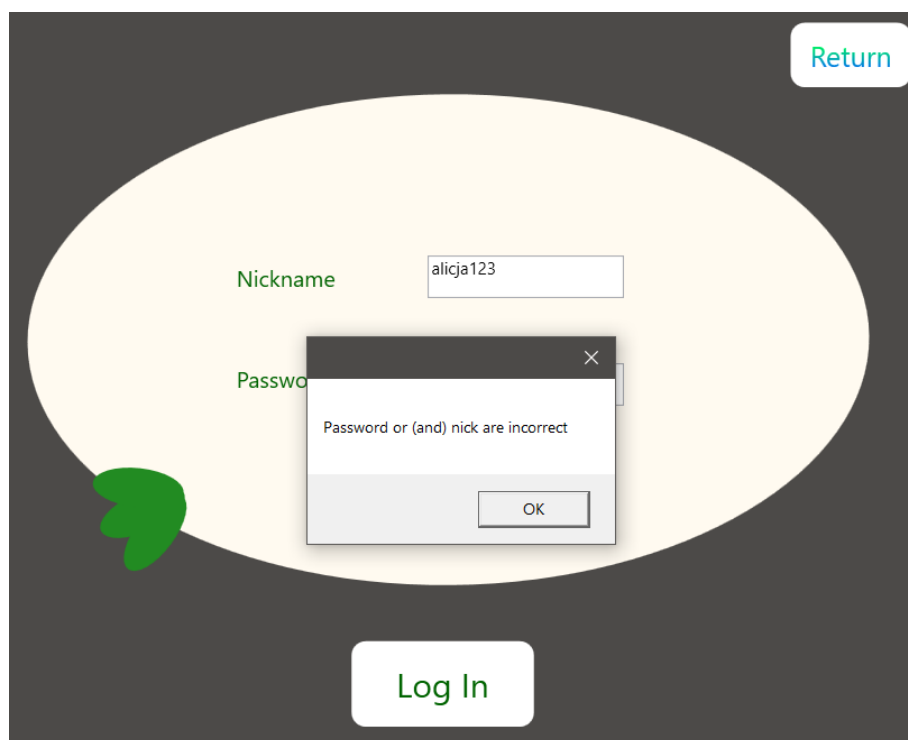
Please, fill any empty spaces left

OK

Rysunek 3: Ostrzeżenie przy rejestracji nowego użytkownika



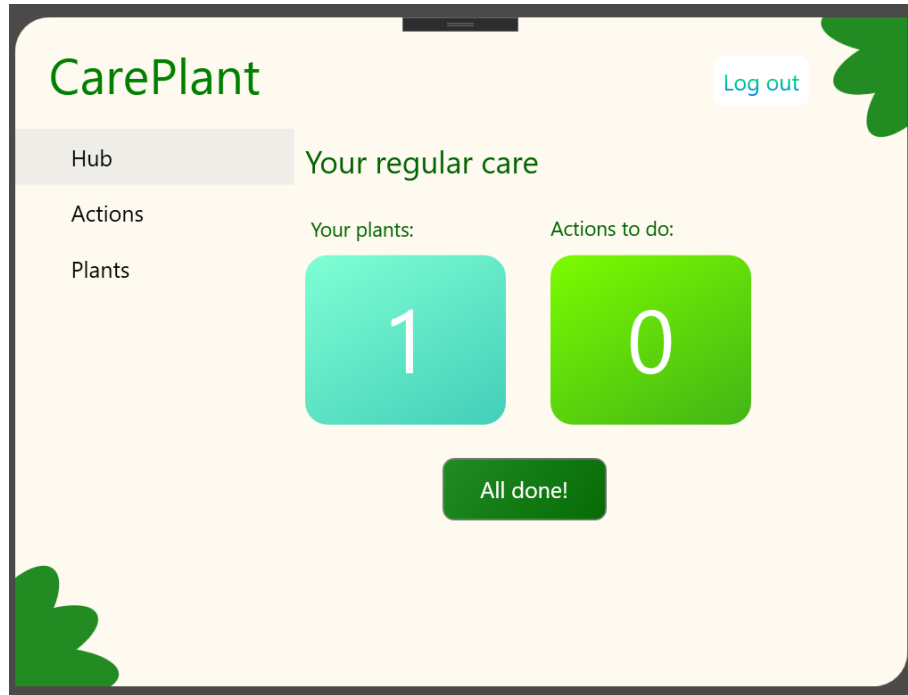
Rysunek 4: Ekran logowania



Rysunek 5: Ostrzeżenie podczas logowania

1.2.2 Obsługa pielęgnacji

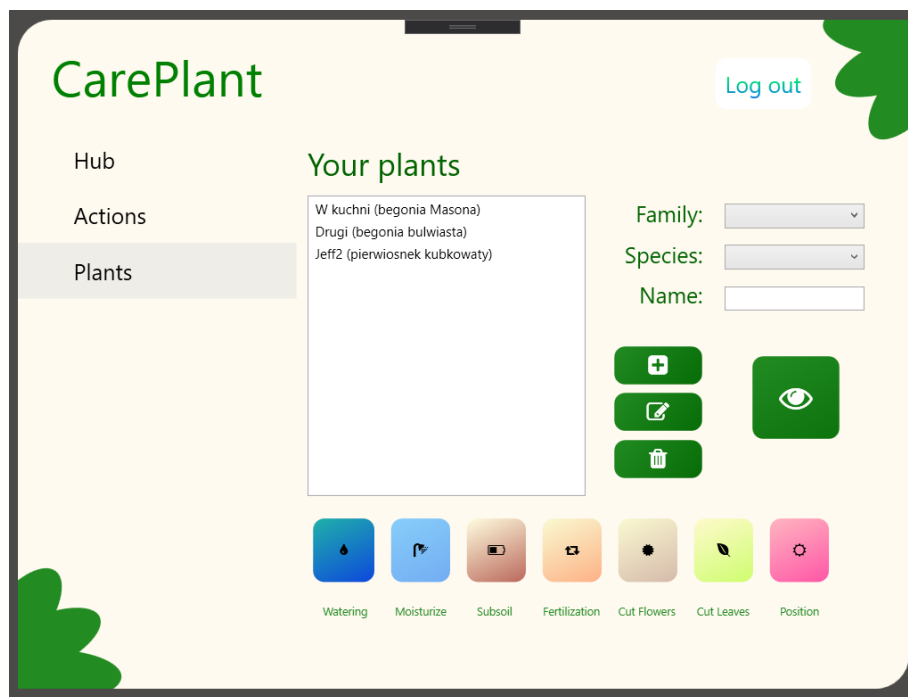
Po zalogowaniu istniejącego użytkownika pojawia się ekran **Hub** z ilością aktualnych powiadomień o nadchodzącej opiece nad roślinami użytkownika. Możliwe jest również zaznaczenie jednym przyciskiem wszystkich akcji jako wykonane, jeśli użytkownik wie, że nie ma nic do zrobienia przy swoich roślinach.



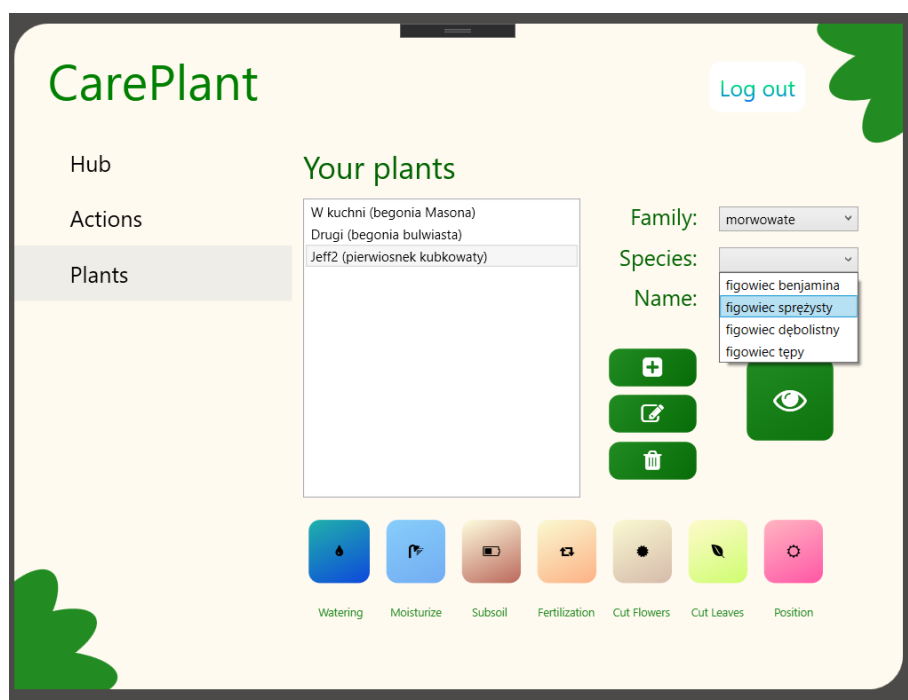
Rysunek 6: Widok ekranu Hub

Po zalogowaniu użytkownik ma do wyboru trzy widoki. Domyślnie pokazuje się widok **Hub** pokazany powyżej.

Zakładka **Plants** pozwala na podgląd wszystkich roślin posiadanych przez użytkownika. Możliwe jest również dodanie, usunięcie lub edytowanie już istniejącego okazu rośliny.

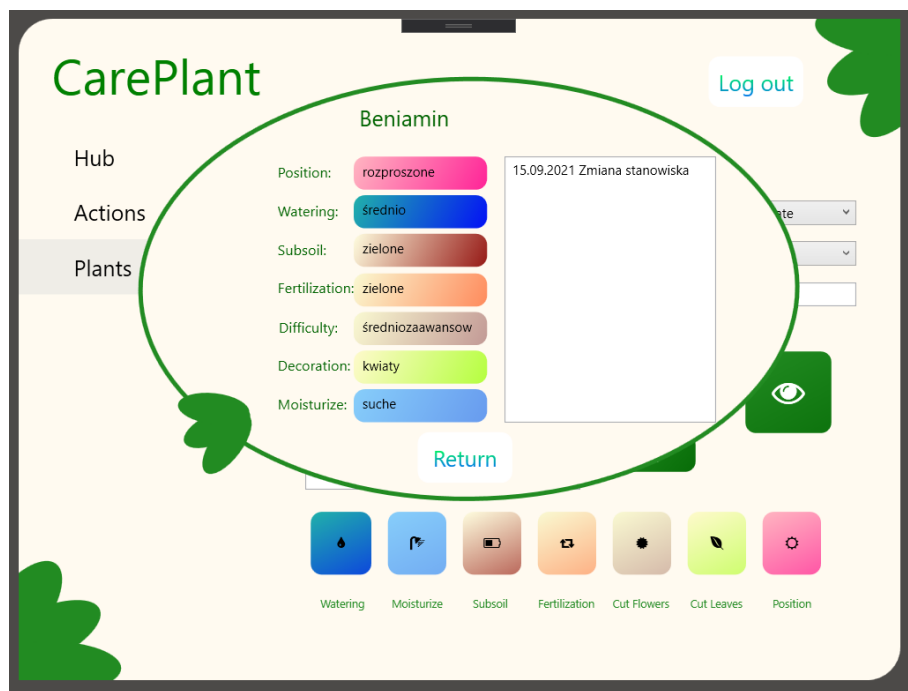


Rysunek 7: Widok ekranu Plants



Rysunek 8: Dodawanie rośliny do zbioru

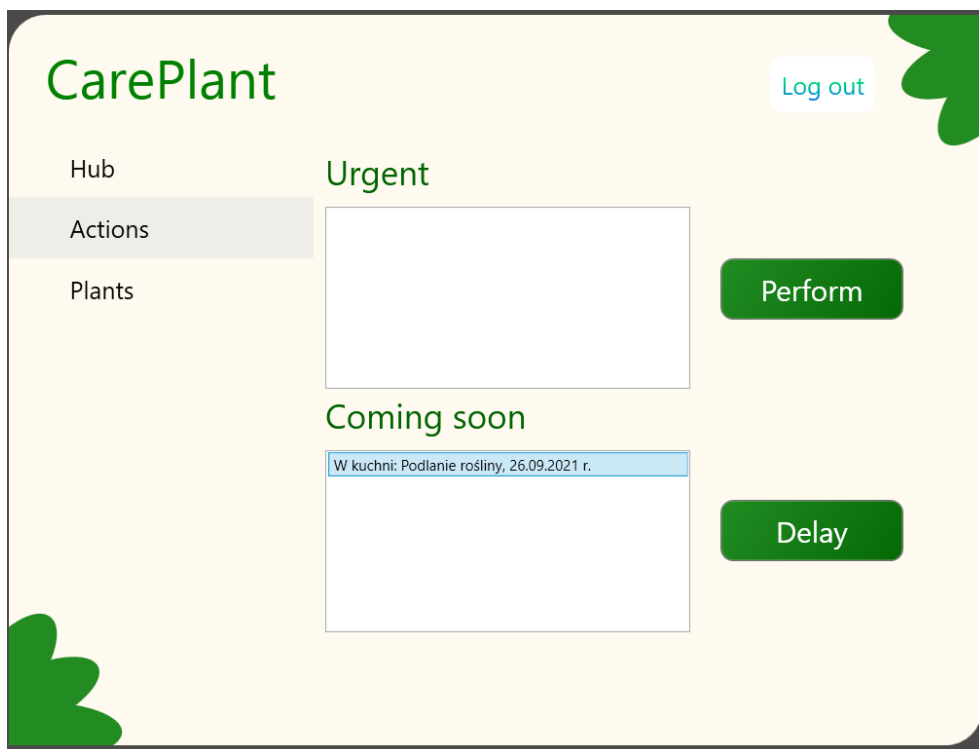
Na dole ekranu znajduje się 7 przycisków, za pomocą których użytkownik rejestruje wykonane akcje na zaznaczonej przez siebie roślinie. Użytkownik po wybraniu konkretnej rośliny ma możliwość podglądu jej wymagań oraz wszystkich akcji, które dotychczas zostały na nim wykonane.



Rysunek 9: Podgląd wymagań i akcji wybranej rośliny

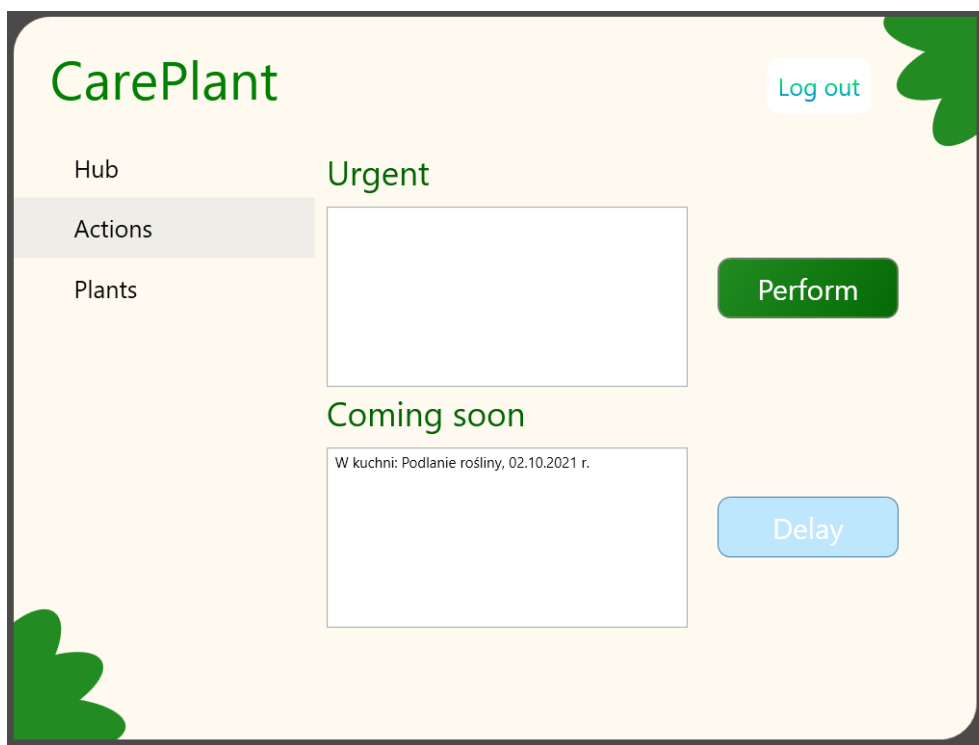
Na trzecim widoku **Actions** widoczne są tworzone przez program przypomnienia o planowanych akcjach, unikalne dla każdej akcji i rośliny należącej do użytkownika. Wyznaczony jest czas wykonania następnej czynności pielęgnacyjnej na podstawie domyślnego odstępu akcji na danym gatunku w połączeniu z algorytmem dostosowującym częstotliwość akcji do potrzeb osoby korzystającej z programu oraz jej członków jej zielonej rodziny. Algorytm dotyczy jednak narazie tylko czynności podlewania roślin

Przypomnienia podzielone są między dwa okna: dla akcji wymagających natychmiastowej interwencji oraz tych, które niedługo nadejdą. Po wybraniu powiadomienia w dowolnym oknie użytkownik ma możliwość zaznaczenia akcji jako wykonanej lub opóźnienia jej w przypadku, gdy na przykład roślina nie wymaga jeszcze podlania.



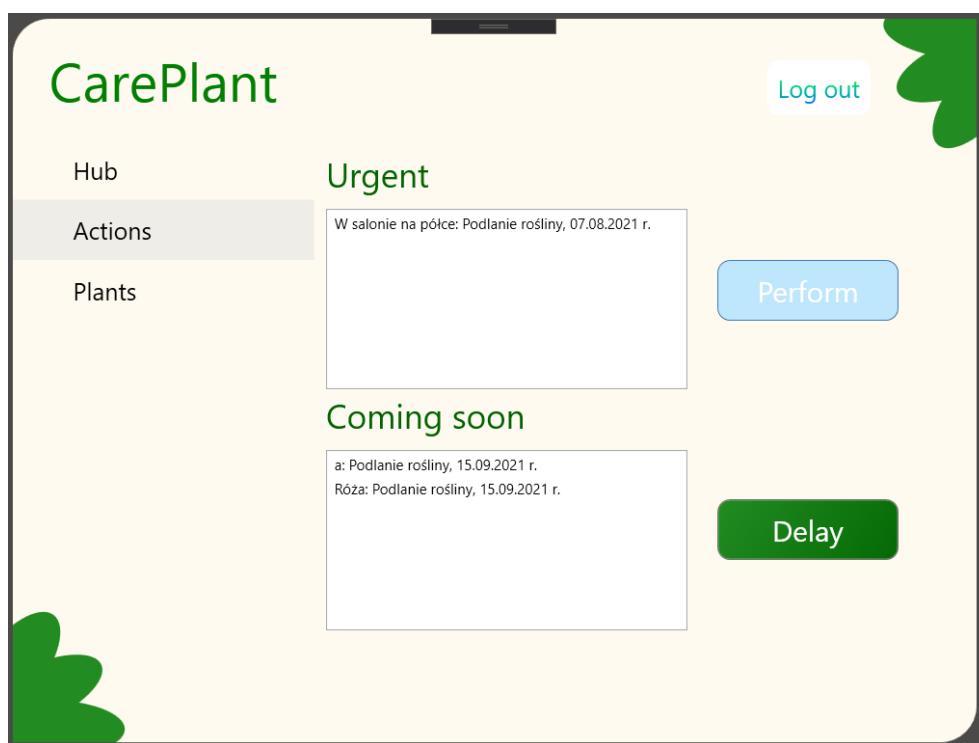
Rysunek 10: Widok przypomnienia o nadchodzącej akcji

Na ekranie widoczna jest akcja do wykonania w oknie nadchodzących akcji. Można opóźnić jej wykonanie, klikając odpowiedni przycisk.



Rysunek 11: Opóźnienie akcji podlewania

Każda wykonana przez użytkownika akcja zostaje zapisana w dzienniku akcji. Po dłuższym czasie użytkownika aplikacji akcje zaczną pojawiać się w oknie akcji pilnych. Można je również wykonać lub opóźnić ich wykonanie.



Rysunek 12: Ekran po wykonaniu akcji podlewania

2 Baza danych

Baza danych została stworzona w celu wspomagania aplikacji Plantcare. Wyszczególnione zostały następujące typy encji:

- RODZINY (nazwa)
- GATUNKI (nazwa, nazwa łacińska)
- OSOBY (login, hasło, imię, nazwisko, wiek)
- ZADANIA (czas wykonania)
- AKCJE (opis akcji)
- DZIENNIKI AKCJI (czas wykonania)
- KTO MA CO (nazwa)
- STANOWISKO (nazwa)
- PODLEWANIE (poziom)

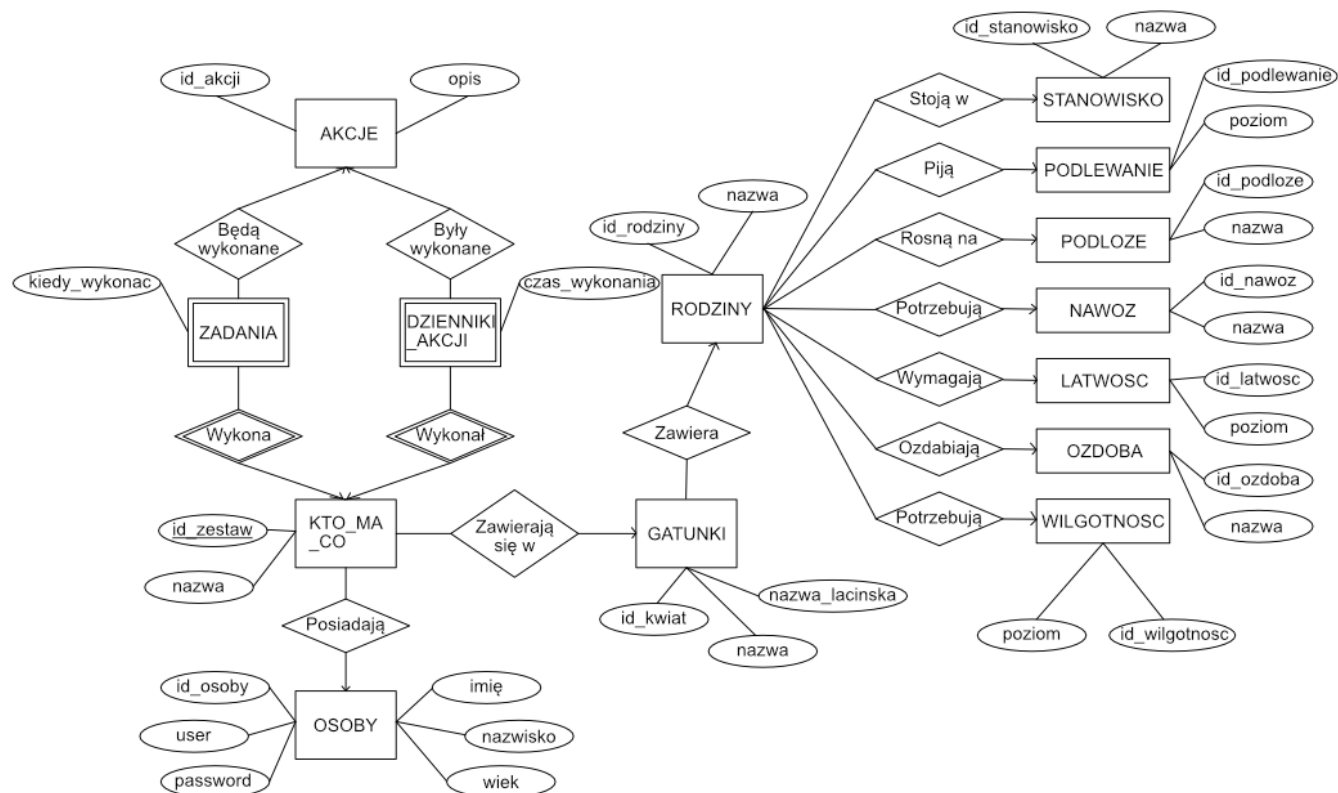
- PODŁOŻE (nazwa)
- NAWÓZ (nazwa)
- ŁATWOŚĆ (poziom)
- OZDOBA (nazwa)
- WILGOTNOŚĆ (poziom)

Tabela RODZINY zawiera spis rodzin roślin doniczkowych, ich cech zewnętrznych oraz informacji niezbędnych do opieki nad nimi, zaś tabela GATUNKI zawiera już sprecyzowane gatunki kwiatów wraz z informacją o przypisanej im rodzinie. W tabeli OSOBY zawarty jest spis zarejestrowanych użytkowników aplikacji.

Całość połączona jest zapisem już zastosowanej oraz planowanej w przyszłości pielęgnacji poszczególnych roślin odpowiednio w tabeli DZIENNIKI AKCJI oraz ZADANIA. Tabela KTO MA CO jest tabelą łączącą właścicieli z ich okazami roślin, jest pomocą w operacjach na tabelach stosowanych do zapisu pielęgnacji.

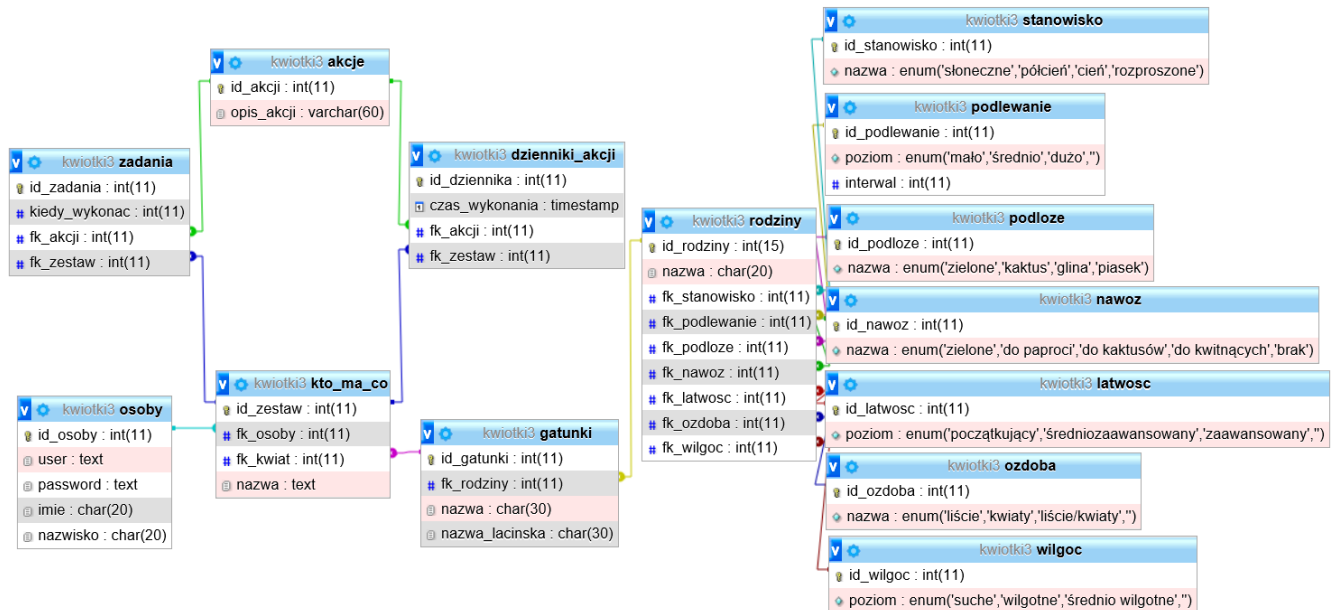
Tabele STANOWISKO, PODLEWANIE, PODŁOŻE, NAWÓZ, ŁATWOŚĆ, OZDOBA, WILGOTNOŚĆ określają możliwości pielęgnacyjne dla każdej z rodzin roślin, mają charakter czysto opisowy.

2.1 Diagram związków encji



Rysunek 13: Diagram związków encji

2.2 Model relacyjny



Rysunek 14: Model relacyjny bazy danych

2.3 Procedury

Na potrzebę aplikacji w bazie przechowywana jest procedura obliczania daty następnego podlania:

```

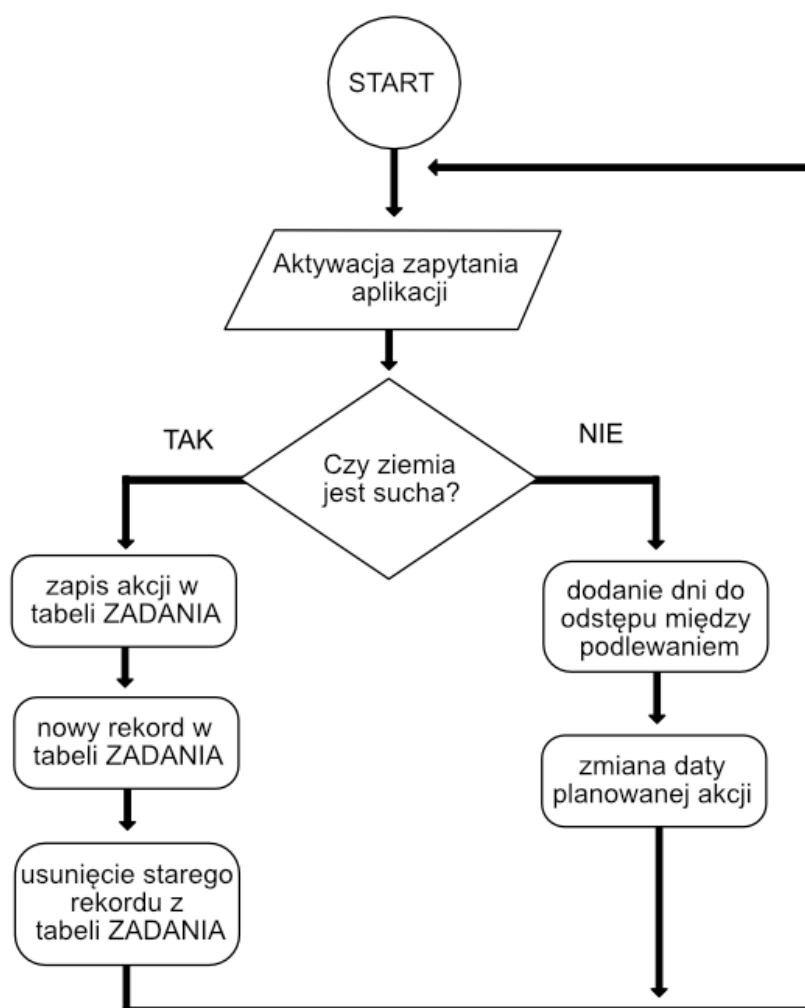
1 BEGIN
2 SELECT d.czas_wykonania, z.kiedy_wykonac
3 INTO @a, @b
4 FROM (dzienniki_akcji d
5       RIGHT JOIN zadania z ON d.fk_akcji = z.fk_akcji
6       AND d.fk_zestaw = z.fk_zestaw)
7 WHERE z.fk_akcji = nr_akcji
8       AND z.fk_zestaw = nr_kwiatka
9       ORDER BY d.czas_wykonania DESC LIMIT 1;
10
11 SELECT @a,
12 (CASE WHEN ISNULL(@a) THEN CURRENT_TIMESTAMP()
13 ELSE @a + INTERVAL @b day
14 END) AS czas_nastepnego;
15 END
  
```

2.4 Algorytm planowanego podlewania

Jak wspomniano wyżej, czas wykonania następnego podlania rośliny wyznaczany jest na podstawie domyślnego odstępu akcji na danym gatunku:

- mało (1) - podlewanie raz na 9 dni
- średnio (2) - podlewanie raz na 4 dni
- dużo (3) - podlewanie raz na 2 dni

Zastosowano również prosty algorytm, który dostosowuje częstotliwość akcji do potrzeb osoby korzystającej z programu na podstawie odpowiedzi użytkownika.



Rysunek 15: Algorytm regulacji odstępów podlewania

Regulacja czasu podlewania zależna jest od wymagań danej rośliny. Dla małego domyślnego podlewania odstęp czasowy zwiększa się o 3 dni, dla średniego o 2, a dla dużego o 1 dzień.

3 Implementacja

Poniżej kilka przykładowych fragmentów kodu, które pokazują istotę rozwiązania problematyki projektu.

- Funkcja w pliku *DataAccess.cs* pozwalająca na edycję rośliny użytkownika w jego bazie danych.

```
1      public bool EditFlower(Flower flower, Species species, String
2          nazwa)
3      {
4          using (connection = new MySqlConnection(connStringBuilder
5              .ToString()))
6          {
7              MySqlCommand command = new MySqlCommand($"UPDATE
8                  kto_ma_co SET nazwa = '{nazwa}', fk_kwiat = {
9                      species.ID} WHERE id_zestaw = {flower.ID} ",
10                  connection);
11              connection.Open();
12              command.ExecuteNonQuery();
13              connection.Close();
14          }
15          return true;
16      }
```

- Komenda w pliku *PlantViewModel.cs* wyzwalająca funkcję edycji powyżej.

```
1      private ICommand _edi = null;
2      public ICommand Edi
3      {
4          get
5          {
6              return _edi ?? (_edi = new BaseClass.RelayCommand(
7                  (p) =>
8                  {
9                      if (currentFlower != null)
10                     {
11                         if (currentFlower.Name != currentName ||
12                             currentFlower.Species !=
13                             currentSpecies)
14                         {
15                             dataAccess.EditFlower(currentFlower,
16                                 currentSpecies, currentName);
17                             Flowers = dataAccess.GetFlowers(
18                                 LogInViewModel.logInfo);
19                         }
20                     }
21                 }
22             },
23             p => true)
24         );
25     }
```

- Komenda w pliku *HubViewModel.cs* wyzwalająca wykonanie wszystkich oczekujących akcji przyciskiem *All done* w widoku *Hub*.

```

1      private ICommand _performAll = null;
2      public ICommand PerformAll
3      {
4          get
5          {
6              return _performAll ?? (_performAll = new BaseClass.
              RelayCommand(
7                  (p) =>
8                  {
9                      foreach (Todo oneTodo in Todos)
10                         oneTodo.Perform();
11                     this.split();
12                 }
13             ,
14             p => true)
15             );
16         }
17     }

```

- Funkcje w pliku *ActionView.xaml.cs* pozwalające na zaznaczenie tylko jednej rośliny jednocześnie przy zamiarze wykonania na niej akcji.

```

1      private void urgentList_Selected(object sender,
2          RoutedEventArgs e)
3      {
4          if(urgentList.SelectedItem != null)
5              soonList.UnselectAll();
6      }
7
8      private void soonList_Selected(object sender, RoutedEventArgs
9          e)
10     {
11         if (soonList.SelectedItem != null)
12             urgentList.UnselectAll();
13     }

```

4 Możliwości rozwoju projektu

- Dodanie zdjęć okazów roślin
- Rozbudowanie algorytmów wyliczających odpowiedni czas pielęgnacji