

**UNIWERSYTET RZESZOWSKI**

**Wydział Nauk Ścisłych i Technicznych**

Łukasz Domin

Nr albumu: 123631

Informatyka I stopnia

**Aplikacja internetowa do zarządzania ogrodem z elementami społecznościowymi**

Praca inżynierska

Praca wykonana pod kierunkiem

….....……………………………………..

Rzeszów, data

Spis treści

[1 Wstęp 3](#_Toc188608806)

[2 Cel i zakres pracy 4](#_Toc188608807)

[3 Projekt aplikacji 5](#_Toc188608808)

[3.1 Narzędzia i technologie użyte w pracy 5](#_Toc188608809)

[3.1.1 Narzędzia użyte do części frontendowej 5](#_Toc188608810)

[3.1.2 Narzędzia użyte do części backendowej 5](#_Toc188608811)

[3.2 Określenie funkcjonalności aplikacji 7](#_Toc188608812)

[3.3 Architektura aplikacji 10](#_Toc188608813)

[4 Implementacja aplikacji 12](#_Toc188608814)

[4.1 Akwizycja danych 12](#_Toc188608815)

[4.2 Struktura bazy danych 13](#_Toc188608816)

[4.2.1 Węzły 13](#_Toc188608817)

[4.2.2 Relacje 21](#_Toc188608818)

[4.2.3 Widok bazy danych 26](#_Toc188608819)

[4.3 Struktura backendu aplikacji 27](#_Toc188608820)

[4.3.1 Przykładowa klasa modelu danych 28](#_Toc188608821)

[4.3.2 Przykład procesu żądania i odpowiedzi 29](#_Toc188608822)

[4.3.3 Logowanie zdarzeń i błędów 33](#_Toc188608823)

[4.3.4 Zadania okresowe w tle 34](#_Toc188608824)

[4.4 Struktura frontendu aplikacji 34](#_Toc188608825)

[4.4.1 Przykładowy komponent 36](#_Toc188608826)

[4.4.2 Przykład metody API z serwisu 38](#_Toc188608827)

[5 Opis funkcjonalności systemu 39](#_Toc188608828)

[5.1 Funkcjonalności zwykłego użytkownika 39](#_Toc188608829)

[5.1.1 Rejestracja 39](#_Toc188608830)

[5.1.2 Zarządzanie kontem użytkownika 40](#_Toc188608831)

[5.1.3 Przeglądanie roślin w katalogu 43](#_Toc188608832)

[5.1.4 Zarządzanie własnymi roślinami 45](#_Toc188608833)

[5.1.5 Zarządzanie postami i komentarzami 45](#_Toc188608834)

[5.1.6 Przeglądanie użytkowników 50](#_Toc188608835)

[5.1.7 Zarządzanie ogrodem 50](#_Toc188608836)

[5.1.8 Zarządzanie rozmowami 55](#_Toc188608837)

[5.1.9 Zarządzanie powiadomieniami 58](#_Toc188608838)

[5.2 Funkcjonalności pracownika 60](#_Toc188608839)

[5.2.1 Zarządzanie roślinami 60](#_Toc188608840)

[5.2.2 Banowanie użytkowników 62](#_Toc188608841)

[5.2.3 Tworzenie specjalnego powiadomienia 63](#_Toc188608842)

[5.3 Funkcjonalności administratora 63](#_Toc188608843)

[5.3.1 Tworzenie konta pracownika 63](#_Toc188608844)

[5.3.2 Banowanie pracowników 64](#_Toc188608845)

[5.3.3 Wysyłanie powiadomień do pracowników 64](#_Toc188608846)

[6 Testy 64](#_Toc188608847)

[7 Podsumowanie 66](#_Toc188608848)

## Wstęp

Aplikacja internetowa pt. „Wirtualny Ogród Yukka” jest systemem umożliwiającym zarządzanie wirtualnym ogrodem, wymianę wiedzy i doświadczeń pomiędzy użytkownikami w przestrzeni społecznościowej.

Tworzona aplikacja dostarcza użytkownikom intuicyjne rozwiązania, takie jak system powiadomień o społecznościowych (np. odpowiedź na komentarz, otrzymanie zaproszenia) lub stricte ogrodowych (np. powiadomienie o okresie owocowania roślin), możliwość projektowania własnych działek z roślinami wybranymi z katalogu, które można następnie dostosować do swoich upodobań. Możliwe jest również stworzenie własnej rośliny we własnym katalogu.

Rośliny oficjalne lub własne mogą być posadzone na swojej działce reprezentowanej w formie siatki kafelków, na których znajduje się roślina oraz obszar jej przydzielony pokolorowany wybranym kolorem, albo z nałożoną przez użytkownika teksturą. Sam widok działki wraz z zasadzonymi na jej roślinami może być udostępniony innym użytkownikom.

Dodatkowo, poprzez integrację części społecznościowej możliwa jest interakcja z innymi użytkownikami, np. poprzez tworzenie postów na różne tematy, które można oceniać lub komentować. Same komentarze też mogą być oceniane i komentowane, pozwalając na aktywną dyskusję pomiędzy członkami aplikacji. Użytkownicy mogą również prowadzić rozmowę między sobą poprzez czat prywatny. W przypadku niewłaściwych zachowań użytkownika, możliwe jest zgłoszenie go przez innych użytkowników i późniejsze zbanowanie go przez pracowników. Utworzona aplikacja wspiera zarówno początkujących, jak i zaawansowanych ogrodników, tworząc przyjazne środowisko do dzielenia się swoją pasją.

## Cel i zakres pracy

Celem aplikacji jest stworzenie interaktywnego narzędzia dla ogrodników, które umożliwia projektowanie własnych wirtualnych ogrodów, zarządzanie roślinami, otrzymywanie powiadomień o zabiegach pielęgnacyjnych bądź o wydarzeniach społecznościowych oraz dzielenie się doświadczeniem i wiedzą z innymi użytkownikami. Aplikacja ma za zadanie ułatwienie planowania i pielęgnacji ogrodu, jednocześnie wspierając społeczność ogrodników, pozwalając na łatwy dialog pomiędzy sobą.

**Zakres pracy**

* Zaprojektowanie i opracowanie struktury bazy danych:
  + stworzenie modelu grafowej bazy danych, dostosowanego do potrzeb aplikacji,
  + ustalenie relacji między węzłami,
  + utworzenie scrappera[[1]](#footnote-1) zbierającego dane do użycia w bazie danych.
* Zaprojektowanie aplikacji:
  + określenie funkcjonalności aplikacji,
  + opracowanie diagramu przypadków użycia dla aplikacji,
  + opracowanie schematów widoków interfejsu użytkownika.
* Tworzenie backendu aplikacji:
  + implementacja komunikacji backendu z bazą danych,
  + implementacja serwerowej strony aplikacji,
  + zaimplementowanie API do komunikacji z frontendem.
* Tworzenie frontendu aplikacji:
  + tworzenie interfejsu użytkownika dla aplikacji,
  + łączenie backendu z frontendem,
  + połączenie API z interfejsem użytkownika, umożliwiająca komunikację między frontendem, a backendem,
  + testowanie i poprawa błędów,
  + przeprowadzenie testów aplikacji, sprawdzających poprawne działanie jej elementów.

## Projekt aplikacji

### Narzędzia i technologie użyte w pracy

#### Narzędzia użyte do części frontendowej

**TypeScript 5.5.2** - Rozszerzenie języka JavaScript dodające typowanie statyczne, klasy oraz interfejsy. Użyty do implementacji części frontendowej aplikacji. Link: <https://www.typescriptlang.org/download/>

**Angular 18.2.11**- Framework oparty na języku TypeScript do budowy aplikacji webowych. Posiada mechanikę sygnałów, wsparcie dla SSR (rendererowania po stronie serwera), hydracji (użycie statycznej części wygenerowanej strony przez SSR), oraz lepsza obsługa ładowania asynchronicznego [1], [2]. Użyty do implementacji części frontendowej aplikacji. Link: <https://v18.angular.dev/installation>

**Bootstrap 5** - biblioteka CSS [3], [4]. Zawiera zestaw narzędzi ułatwiających tworzenie interfejsu aplikacji internetowych. Bazuje na gotowych rozwiązaniach HTML oraz CSS i może być stosowany m.in. do stylizacji poszczególnych elementów strony. Użyty do implementacji frontendowej aplikacji. Link: <https://getbootstrap.com>

**Npm (Node Package Manager**) - Menedżer pakietów dla JavaScriptu, dostarczany wraz z Node.js. Służy do zarządzania bibliotekami oraz narzędziami JavaScript. Link: <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

#### Narzędzia użyte do części backendowej

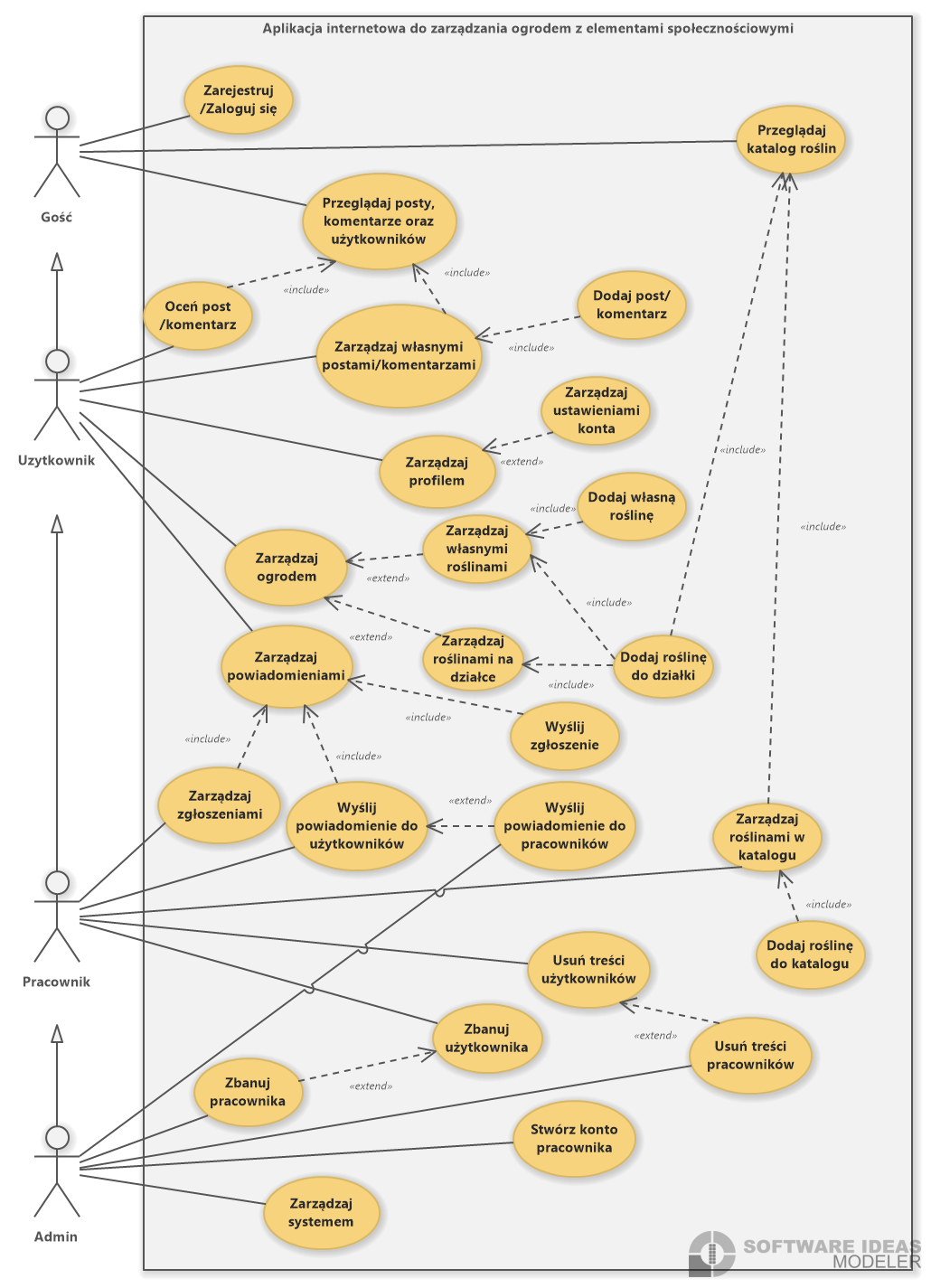
**Java 17** - Język obiektowy działający na różnych systemach operacyjnych bez większych trudów. Jest szeroko stosowana do tworzenia przeróżnych aplikacji czy to internetowych, czy to mobilnych [5]. Użyty do implementacji części backendowej aplikacji. Link: <https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>

**Spring Boot 3.3.2** - Framework języka Java, który upraszcza proces konfiguracji i uruchamiania aplikacji, automatycznie konfigurując większość komponentów, skracając kod przy użyciu adnotacji, a także upraszcza komunikację z bazą danych i frontendem [6], [7]. Użyty do implementacji części backendowej aplikacji. Link: <https://docs.spring.io/spring-boot/installing.html>

**Neo4j 5.12.0** – Jest to grafowa baza danych, która przechowuje dane w postaci węzłów, ich etykiet, relacji pomiędzy nimi oraz właściwościami węzłów bądź relacji. Została ona wybrana dlatego, że ten typ bazy danych jest polecany w budowaniu aplikacji społecznościowych ze względu na budowę relacji pomiędzy poszczególnymi elementami bazy [8], [9].

### Określenie funkcjonalności aplikacji

W aplikacji przewidziano trzy role użytkowników: Użytkownik, Pracownik i Admin. Każda z tych ról dysponuje odpowiednim zakresem funkcjonalności, zostały przedstawione na rysunku 1, tj. diagramie przypadków użycia dla aplikacji [10], [11].



Rysunek Diagram przypadków użycia

**Definicja aktorów**

**AKTOR:** Gość

**OPIS:** Gość może przeglądać katalog roślin, posty, komentarze, profile użytkowników i może się zarejestrować lub zalogować.

**PRZYPADKI UŻYCIA:**

* PU Zarejestruj/Zaloguj się
* PU Przeglądaj posty, komentarze oraz użytkowników
* PU Przeglądaj katalog roślin

**AKTOR:** Użytkownik

**OPIS:** Użytkownik oprócz funkcjonalności gościa oceniać posty i komentarze oraz tworzyć własne i nimi zarządzać. Może również zarządzać kontem, swoimi powiadomieniami roślinami własnymi, ogrodem i działkami wewnątrz niego, a także zgłaszać innych użytkowników.

**PRZYPADKI UŻYCIA:**

* PU Zarządzaj własnymi postami/komentarzami
* PU Oceń post/komentarz powiązane przez <<include>> z PU Przeglądaj posty, komentarze oraz użytkowników
* PU Zarządzaj własnymi postami/komentarzami powiązane przez <<include>> z PU Przeglądaj posty, komentarze oraz użytkowników
* PU Dodaj post/ komentarz powiązane przez <<include>> z PU Zarządzaj własnymi postami/komentarzami
* PU Zarządzaj profilem
* PU Zarządzaj ustawieniami konta powiązane przez <<extend>> z PU Zarządzaj profilem
* PU Zarządzaj ogrodem
* PU Zarządzaj własnymi roślinami powiązane przez <<extend>> z PU Zarządzaj ogrodem
* PU Dodaj własną roślinę powiązane przez <<include>> z PU Zarządzaj własnymi roślinami
* PU Zarządzaj roślinami na działce powiązane przez <<extend>> z PU Zarządzaj ogrodem
* PU Dodaj roślinę do działki powiązane przez <<include>> z PU Zarządzaj roślinami na działce, PU Zarządzaj własnymi roślinami oraz PU Przeglądaj katalog roślin
* PU Zarządzaj powiadomieniami
* PU Wyślij zgłoszenie powiązane przez <<include>> z PU Zarządzaj powiadomieniami

**AKTOR:** Pracownik

**OPIS:** Pracownik oprócz funkcjonalności użytkownika zarządza roślinami w katalogu, usuwa niestosowne treści, przegląda zgłoszenia i banuje.

**PRZYPADKI UŻYCIA:**

* PU Zarządzaj zgłoszeniami powiązane przez <<include>> z PU Zarządzaj powiadomieniami
* PU Wyślij powiadomienie do użytkowników powiązane przez <<include>> z PU Zarządzaj powiadomieniami
* PU Zarządzaj roślinami w katalogu powiązane przez <<include>> z PU Przeglądaj katalog roślin
* PU Dodaj roślinę do katalogu powiązane przez <<include>> z PU Zarządzaj roślinami w katalogu
* PU Usuń treści użytkowników
* PU Zbanuj użytkownika

**AKTOR:** Administrator

**OPIS:** Administrator oprócz funkcjonalności pracownika zarządza systemem oraz pracownikami, wysyła im powiadomienia, usuwa ich treści, banuje, a także tworzy nowe konta pracowników.

**PRZYPADKI UŻYCIA:**

* PU Wyślij powiadomienie do pracowników powiązane przez <<extend>> z PU Wyślij powiadomienie do użytkowników
* PU Zbanuj pracownika powiązane przez <<include>> z PU Zbanuj użytkownika
* PU Usuń treści pracowników powiązane przez <<extend>> z PU Usuń treści użytkowników
* PU Stwórz konto pracownika
* PU Zarządzaj systemem

**Przykładowy przypadek użycia**

**PU ZAREJESTRUJ/ZALOGUJ SIĘ**

**OPIS**

**CEL:** ZALOGOWANIE DO SYSTEMU

**WS:** może być wywołany z PU Załóż konto/Rejestruj

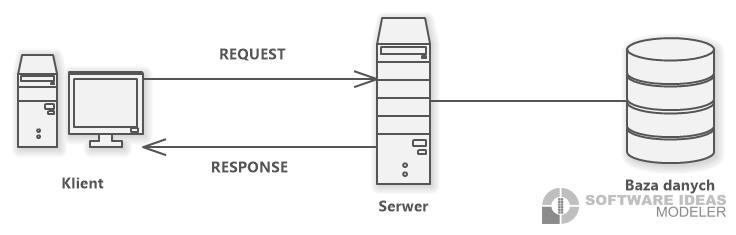
**WK:** podanie niezbędnych danych do logowania tj. login i hasło

**PRZEBIEG:**

Do logowania należy podać e-mail/nazwę użytkownika i hasło. W razie błędu wyświetla komunikat. Po poprawnym logowaniu użytkownik trafia na stronę główną. Rejestracja wymaga podania nazwy, e-maila, hasła i jego potwierdzenia. Jeśli e-mail lub nazwa jest zajęta, pojawia się stosowny komunikat. Po poprawnej rejestracji na e-mail wysyłany jest kod weryfikacyjny do aktywacji konta.

### Architektura aplikacji

Diagram (zob. rysunek 2) przedstawia architekturę aplikacji internetowej modelu klient-serwer [12].



Rysunek Widok architektury aplikacji

Architektura aplikacji składa się z trzech warstw:

Klient (Frontend) – działa na aplikacji opracowanej we frameworku Angular, która komunikuje się z serwerem backendowym przy użyciu żądań HTTP, wysyłając je do REST API[[2]](#footnote-2) i otrzymując odpowiedzi [13].

Serwer (Backend) – utworzony we frameworku Spring Boot, odpowiada za obsługę żądań z frontendu i komunikację z bazą danych, wysyłając odpowiedzi do klienta.

Baza danych – grafowa baza danych Neo4j,przyjmuje zapytania od backendu w języku CYPHER[[3]](#footnote-3), odczytując lub modyfikując dane i zwracając je.

## Implementacja aplikacji

### Akwizycja danych

Głównym źródłem danych dla aplikacji jest strona <https://e-katalogroslin.pl> należąca do Związku Szkółkarzy Polskich będącym stowarzyszeniem producentów drzew, krzewów i innych roślin. Strona jest katalogiem zawierającym ponad 10 000 roślin o różnych właściwościach takich jak rodzaj gleby, zimozieloność liści, zakres wysokości itp.. W celu akwizycji tych danych do bazy danych aplikacji, utworzono odpowiednie oprogramowanie tj. scrapper napisany w języku Python [14], który pobiera dane do pliku *katalog\_roslin.csv.* Wykorzystuje sterownik Firefox, wielowątkowość oraz asynchroniczność do szybszego pobierania danych ze stron. Aby rozpocząć proces pobierania roślin, należy uruchomić plik *scrap\_threads.py* pobierający dane od początkowej strony aż do ostatniej. Plik *katalog\_roslin.csv* zawiera informacje o roślinach tj.: nazwa rośliny, nazwa łacińska, opis, rodzaj gleby, a także link do obrazu rośliny (jeśli takowy posiada). Aby pobrać obrazy, należy uruchomić plik *download\_images.py,* który iteruje przez kolejne rzędy pliku *katalog\_roslin.csv* i zapisuje obrazy do folderu *images.*

Na rysunku 3, pokazano fragment kodu źródłowego scrappera, tj. pliku *scrap\_threads.py*. Fragment kodu przedstawia funkcję *get\_all\_plant\_info*, która pobiera informacje o roślinach z listy adresów URL. Używa ona biblioteki *Selenium* do nawigacji po stronach internetowych i oczekiwania na załadowanie poszczególnych elementów. Po przetworzeniu strony następuje przejście do następnej. Funkcja zwraca dane w postaci listy zawierającej rośliny wraz z niezbędnymi informacjami. Na koniec zamyka przeglądarkę. Warto wspomnieć, że aplikacja scrappera działa wielowątkowo, przydzielając zakres adresów do przetworzenia dla każdego wątku. Po przetworzeniu wszystkich wątków, zwrócone dane z poszczególnych wątków zapisywane są do pliku *katalog\_roslin.csv*.



Rysunek Funkcja get\_all\_plant\_info z pliku scrap\_threads.py

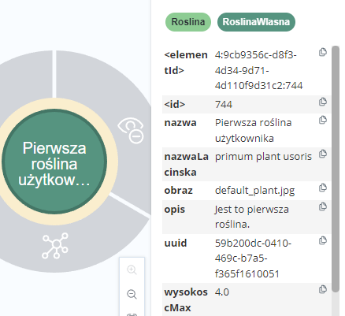
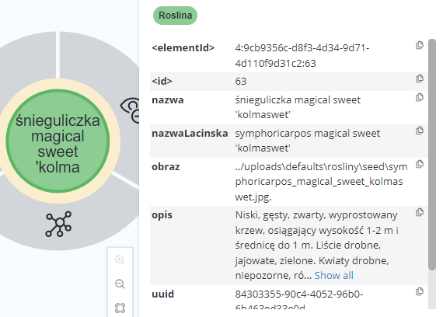
### Struktura bazy danych

#### Węzły

W strukturze bazy danych przewidziano następujące typy węzłów: *Roslina*, *RoslinaWlasna*, *Cecha*, *Ogrod*, *Dzialka*, *Uzytkownik*, *Pracownik,* *Admin*, *Ustawienia*, *Post*, *Komentarz*, *RozmowaPrywatna*, *Powiadomienie*, Zgloszenie. Niektóre z tych węzłów mają swoje podtypy tzn. są one podwójnie etykietowane, np. Cecha roślin ma dodatkowe etykiety tj. *Forma*, *Gleba*, *Owoc* itp. Poniżej dokonano krótkiej charakterystyki każdego z typów węzłów.

###### Roslina, RoslinaWlasna

*Roslina* (zob. rys. 4) jest jednym z głównym węzłów w bazie danych. Reprezentuje on roślinę z katalogu oraz roślinę użytkownika, które użytkownik może wykorzystać projektując ogród. Rośliny własne, czyli te dodane przez użytkownika, nienależące do oficjalnego katalogu posiadają etykietę *Roslina* oraz *RoslinaWlasna*, a roślina z katalogu ma tylko etykietę *Roslina*.



Rysunek Węzeł Roslina i węzeł Roslina z etykietą RoslinaWlasna

Właściwości węzłów Roslina, RoslinaWlasna:

* *uuid* – identyfikator rośliny,
* *nazwa* – nazwa rośliny,
* *nazwaLacinska* – nazwa łacińska rośliny,
* *opis* – opis rośliny,
* *wysokoscMin, wysokoscMax* – wysokość minimalna i maksymalna rośliny,
* *obraz* – ścieżka do obrazu rośliny, jeśli takowy posiada.

###### Cecha

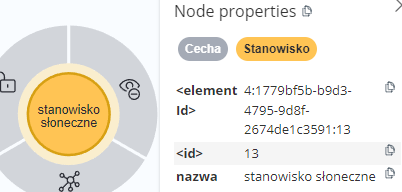
Rośliny posiadają 16 cech opisanych etykietami (zob. tabela 1). Etykiety te stanowią dodatkowe określenie cechy roślin, zatem jak widać, każdy węzeł z etykietą *Cecha* zawiera również dodatkową etykietę doprecyzowującą, uszczegóławiającą tj. *Cecha:Forma, Cecha:Owoc,* itp. które klasyfikują typ cechy. Niektóre rośliny posiadają również dodatkowo trzecią etykietę tj. *CechaWlasna*. Oznacza to, że użytkownik dodał jeszcze własną cechę, którą może przypisać tylko i wyłącznie do własnej rośliny (zob. rys. 5).

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Opis** | **Widok węzła** |
| Forma | Forma rośliny, np. krzewinka, bylina, itp. |  |
| Gleba | Rodzaj gleby przyjazny dla rośliny, np. gliniasta, piaszczysta, itp. |  |
| Grupa | Grupa rośliny, np. róże, krzewy, itp. |  |
| Kolor | Kolor kwiatów albo liści rośliny, np. czerwone, białe, itp. |  |
| Kwiat | Rodzaj kwiatu rośliny, np. pełne, pączkowe, itp. |  |
| Odczyn | Odczyn ph gleby przyjazny dla rośliny, np. kwaśny, zasadowy, itp. |  |
| Okres | Okres owocowania albo kwitnienia rośliny wyrażony w miesiącach |  |
| Owoc | Rodzaj owocu rośliny, np. kolczaste, strąki, itp. |  |
| Podgrupa | Podgrupa rośliny, np. drzewa owocowe, wrzosy, itp. |  |
| Pokroj | Pokrój lub kształt rośliny, np. stożkowy, jajowaty, itp. |  |
| SilaWzrostu | Siła wzrostu rośliny, czyli jak szybko rośnie. |  |
| Stanowisko | Odpowiednie nasłonecznienie rośliny, np. słoneczne, cieniste, itp. |  |
| Wilgotnosc | Optymalna wilgotność gleby rośliny, np. podłoże wilgotne, podmokłe, itp. |  |
| Zastosowanie | Do czego pasuje roślina albo czy ma jakiś użytek poza wyglądem |  |
| Zimozielonosc | Jak zachowują się liście/igły rośliny podczas zimy, czy opadają, czy nie) |  |

Tabela Zestawienie dodatkowych cech roślin

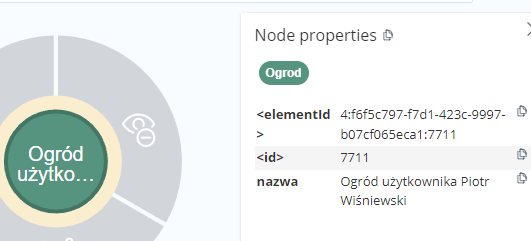
Właściwości węzłów *Cecha*, *CechaWlasna*:

* *nazwa* – nazwa cechy.



Rysunek Węzeł Cecha i węzeł Cecha z etykietą CechaWlasna

###### Ogrod

Ogród (zob. rys. 6) jest węzłem tworzonym dla każdego użytkownika. Reprezentuje ogród połączony z dziesięcioma węzłami działek należących do użytkownika. W węźle Ogrod użytkownik definiuje nazwę własnego ogrodu.

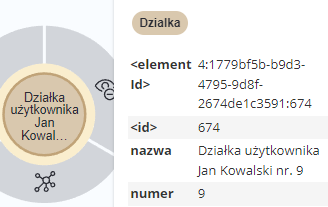
Właściwości węzła *Ogrod*:

* *nazwa* – nazwa ogrodu.

Rysunek Węzeł ogrodu

###### Dzialka

Dzialka (zob. rys. 7) jest węzłem związanym z ogrodem i tworzonym wraz z nim. Łącznie dla każdego użytkownika jest tworzone po 10 działek na ogród, gdzie każda działka ma wymiary 20x20 kafelków.

Właściwości węzła *Dzialka*:

* *nazwa* – nazwa działki,
* *numer* – numer działki od 1 do 10.

Rysunek Węzeł Dzialka

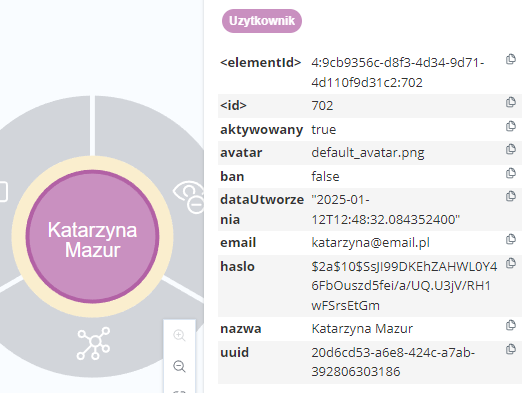
###### Uzytkownik, Pracownik, Admin

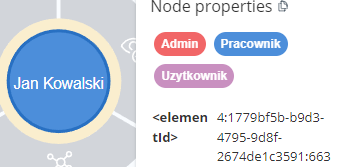
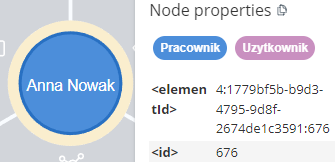
Węzeł *Uzytkownik* (zob. rys. 8) jest węzłem pełniącym rolę użytkownika w bazie danych. Zwykli użytkownicy posiadają tylko etykietę *Uzytkownik*, natomiast pracownicy posiadają dodatkowo etykietę *Pracownik*, z kolei administratorzy posiadają wszystkie z tych etykiet oraz dodatkowo etykietę *Admin*. Każda etykieta oznacza inny poziom uprawnień zdefiniowany w rozdziale 3.2 Określenie funkcjonalności aplikacji.

Dodatkowo, każdy użytkownik ma połączenia z węzłem *Ustawienia*, węzłem *Ogrod* oraz 10 węzłami *Dzialka* przypisanych do tego ogrodu. Informacje o użytkowniku rozdzielone są w taki sposób, aby zoptymalizować pobór danych z bazy i pobierać tylko te części danych, które są aktualnie potrzebne.

Właściwości węzłów *Uzytkownik*, *Pracownik*, *Admin*:

* *uuid* – identyfikator użytkownika,
* *nazwa* – nazwa użytkownika,
* *email* – adres email użytkownika,
* *haslo* – zaszyfrowane hasło użytkownika,
* *avatar* – zdjęcie użytkownika,
* *dataUtworzenia* – data utworzenia użytkownika,
* *aktywowany* – oznacza, czy konto zostało aktywowane po rejestracji,
* *ban* – czy użytkownik jest zbanowany,
* *banDo* – data, do której użytkownik jest zbanowany,
* *banPowod* – powód zbanowania,
* *imie* – imię użytkownika na jego profilu,
* *nazwisko* – nazwisko użytkownika na jego profilu,
* *miasto* – miasto użytkownika na jego profilu,
* *miejsceZamieszkania* – miejsce zamieszkania użytkownika na jego profilu,
* *opis* – opis użytkownika na jego profilu.





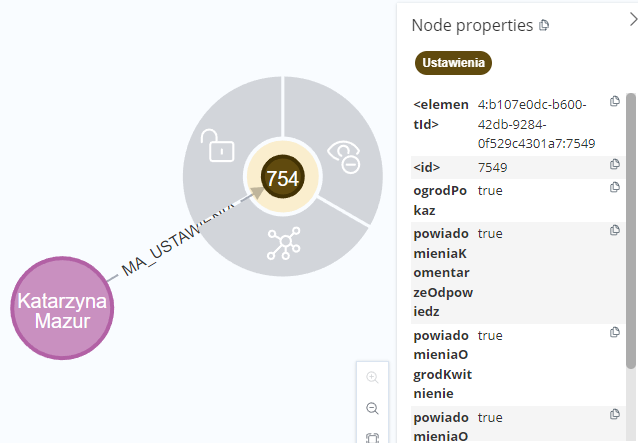
Rysunek Węzeł Uzytkownik oraz węzły z etykietami Pracownik i Admin

###### Ustawienia

Ustawienia (zob. rys. 9) definiują ustawienia użytkownika, w których decyduje się o powiadomieniach i widoczności swojego profilu.

Właściwości węzła *Ustawienia*:

* *ogrodPokaz* – pole umożliwiające pokazanie lub ukrycie ogrodu (nie dotyczy pracowników i administratorów),
* *powiadomieniaKomentarzeOdpowiedz* – pole ustawiające możliwość powiadamiania o odpowiedziach na komentarze,
* *powiadomieniaOgrodKwitnienie* – pole ustawiające możliwość powiadamiania o kwitnieniu roślin w ogrodzie,
* *powiadomieniaOgrodOwocowanie* – pole ustawiające możliwość powiadamiania o owocowaniu roślin,
* *statystykiProfilu* – pole ustawiające widoczność statystyk na profilu.



Rysunek Węzeł Ustawienia

###### Post

Węzeł *Post* (zob. rys. 10) przechowuje posty dla funkcjonalności społecznościowych w aplikacji. Oprócz nazwy i opisu może również zawierać obraz dołączony do postu. Do postu mogą być przypisane komentarze użytkowników w formie osobnych węzłów *Komentarz*, a także oceny w formie relacji *OCENIL*.

Właściwości węzła *Post*:

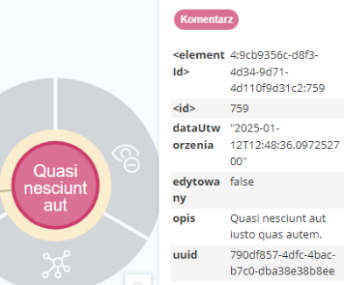
* *uuid* – identyfikator postu
* *tytul* – tytuł postu
* *opis* – opcjonalny opis postu
* *obraz* – ścieżka do obrazu postu
* *dataUtworzenia* – data utworzenia postu



Rysunek Węzeł Post

###### Komentarz

Węzeł *Komentarz* (zob. rys. 11) przechowuje komentarze dla funkcjonalności społecznościowych. Są one używane do komentowania postów, odpowiadania na komentarze i mogą być oceniane. Dodatkowo służą także jako pojedyncze wiadomości w rozmowach prywatnych.

Właściwości węzła *Komentarz*:

* *uuid* – identyfikator komentarza,
* *opis* – treść komentarza,
* *obraz* – ścieżka do obrazu,
* *edytowany* – informacja, czy komentarz był edytowany,
* *dataUtworzenia* – data utworzenia.

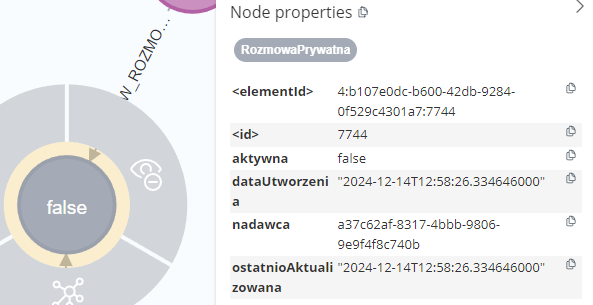
Rysunek Węzeł Komentarz

###### RozmowaPrywatna

Węzeł *RozmowaPrywatna* (zob. rys. 12) przechowuje rozmowy prywatne pomiędzy dwoma użytkownikami w części społecznościowej. Użytkownik *A* musi zaprosić użytkownika *B* do rozmowy, a odbiorca musi ją zaakceptować. Wtedy możliwe jest pisanie wiadomości pomiędzy nimi. Poszczególne treści rozmów przechowywane są w węzłach *Komentarz* połączonych z rozmową relacją *MA\_WIADOMOSC*.

Właściwości węzła *RozmowaPrywatna*:

* *aktywna* – czy rozmowa prywatna jest aktywna (odbiorca zaakceptował zaproszenie),
* *nadawca* – identyfikator nadawcy,
* *dataUtworzenia* – data utworzenia rozmowy,
* *ostatnioAktualizowana* – data zmiany.



Rysunek Węzeł RozmowaPrywatna

###### Powiadomienie, Zgloszenie

Węzeł *Powiadomienie* (zob. rys. 13) powiadania użytkownika o różnych zdarzeniach w systemie, tj. odpowiedź na komentarz, rozpoczęcie okresu owocowania roślin na działce czy zaproszeniu przez użytkownika do rozmowy. Jeśli posiada ono dodatkowo etykietę *Zgloszenie*, to pełni ono rolę zgłoszenia przez użytkownika, które jest przypisywane do pracownika o najmniejszej aktualnej liczbie zgłoszeń. Możliwe jest ustawienie powiadomienia jako przeczytane, albo jego usunięcie pod warunkiem, że nie jest zgłoszeniem.

Właściwości węzłów *Powiadomienie*, *Zgloszenie*:

* *typ* – rodzaj powiadomienia (komentarz\_post, zaproszenie, owocowanie\_roslin),
* *opis* – wygenerowany opis powiadomienia,
* *odnośnik* – odnośnik do elementu wspomnianego w opisie (post, rozmowa prywatna, ogród),
* *dataUtworzenia* – data utworzenia powiadomienia,
* *avatar* – ścieżka do obrazu danego powiadomienia.

Rysunek Węzeł Powiadomienie bez i z etykietą Zgloszenie

#### Relacje

W strukturze bazy danych przewidziano następujące typy relacji: STWORZONA\_PRZEZ, SKOMENTOWAL, ODPOWIEDZIAL, JEST\_W\_POSCIE, JEST\_W\_ROZMOWIE, itp. Niektóre z tych relacji mają swoje właściwości tzn. oprócz identyfikatora posiadają cechy podobnie jak w węzłach. Poniżej dokonano krótkiej charakterystyki każdego typu relacji z podziałem na relacje bez właściwości i z właściwościami.

###### Relacje bez właściwości

W tabeli 2 przedstawiono wszystkie relacje bez właściwości w bazie danych. Wskazują one tylko typy połączonych węzłów oraz kierunek relacji.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Kierunek** | **Opis** | **Przykład** |
| STWORZONA\_PRZEZ | Uzytkownik  |  V  RoslinaWlasna | Wskazuje roślinę nienależącą do katalogu stworzoną przez użytkownika. |  |
| SKOMENTOWAL | Uzytkownik  |  V  Komentarz | Wskazuje komentarz, który utworzył użytkownik. |  |
| ODPOWIEDZIAL | Komentarz  |  V  Komentarz | Wskazuje komentarz, na który odpowiedział użytkownik. |  |
| JEST\_W\_POSCIE | Komentarz  |  V  Post | Wskazuje post zawierający wybrany komentarz. |  |
| JEST\_W\_ROZMOWIE | Uzytkownik  |  V  RozmowaPrywatna | Wskazuje rozmowę, w której jest użytkownik. |  |
| MA\_KOMENTARZ | Post  |  V  Komentarz | Wskazuje komentarz komentujący dany post. |  |
| MA\_POST | Uzytkownik  |  V  Post | Wskazuje post stworzony przez użytkownika. |  |
| MA\_WIADOMOSC | RozmowaPrywatna  |  V  Komentarz | Wskazuje komentarz w rozmowie. |  |
| WYSYLA\_POWIADOMIENIE | Uzytkownik  |  V  Powiadomienie | Wskazuje powiadomienie globalne wysyłane przez użytkownika. |  |
| ZGLASZA | Uzytkownik  |  V  Powiadomienie | Wskazuje zgłoszenie, które wysłał użytkownik do pracownika. |  |
| MA\_USTAWIENIA | Uzytkownik  |  V  Ustawienia | Wskazuje ustawienia użytkownika. |  |
| MA\_OGROD | Uzytkownik  |  V  Ogrod | Wskazuje ogród użytkownika. |  |
| MA\_DZIALKE | Ogrod  |  V  Dzialka | Wskazuje działkę ogrodu użytkownika. |  |
| MA\_FORME | Roslina,  RoslinaWlasna  |  V  Forma | Wskazuje formę rośliny. |  |
| MA\_GLEBE | Roslina,  RoslinaWlasna  |  V  Gleba | Wskazuje stosowną glebę dla rośliny. |  |
| MA\_GRUPE | Roslina,  RoslinaWlasna  |  V  Grupa | Wskazuje grupę rośliny. |  |
| MA\_KOLOR \_KWIATOW | Roslina,  RoslinaWlasna  |  V  Kolor | Wskazuje kolor kwiatów rośliny. |  |
| MA\_KOLOR\_LISCI | Roslina,  RoslinaWlasna  |  V  Kolor | Wskazuje kolor liści rośliny. |  |
| MA\_OKRES \_KWITNIENIA | Roslina,  RoslinaWlasna  |  V  Okres | Wskazuje miesiące, w której roślina kwitnie. |  |
| MA\_OKRES\_OWOCOWANIA | Roslina,  RoslinaWlasna  |  V  Okres | Wskazuje miesiące, w której roślina owocuje. |  |
| MA\_KWIAT | Roslina,  RoslinaWlasna  |  V  Kwiat | Wskazuje rodzaj kwiatów, jakie ma roślina. |  |
| MA\_ODCZYN | Roslina RoslinaWlasna  |  V  Odczyn | Wskazuje odczyn gleby, w jakich roślina powinna być sadzona. |  |
| MA\_OWOC | Roslina, RoslinaWlasna  |  V  Owoc | Wskazuje rodzaj, kształt albo kolor owocu rośliny. |  |
| MA\_PODGRUPE | Roslina, RoslinaWlasna  |  V  Podgrupa | Wskazuje podgrupę, do której należy roślina. |  |
| MA\_POKROJ | Roslina, RoslinaWlasna  |  V  Pokroj | Wskazuje pokrój rośliny. |  |
| MA\_SILE\_WZROSTU | Roslina, RoslinaWlasna  |  V  SilaWzrostu | Wskazuje szybkość rośnięcia rośliny. |  |
| MA\_ZIMOZIELONOSC\_LISCI | Roslina, RoslinaWlasna  |  V  Zimozielonosc | Wskazuje zachowanie liści podczas zimy. |  |
| MA\_WILGOTNOSC | Roslina, RoslinaWlasna  |  V  Wilgotnosc | Wskazuje optymalną wilgotność gleby dla rośliny. |  |
| MA\_STANOWISKO | Roslina, RoslinaWlasna  |  V  Stanowisko | Wskazuje optymalne stanowisko rośliny. |  |
| MA\_WALOR | Roslina, RoslinaWlasna  |  V  Walor | Wskazuje walory rośliny. |  |
| MA\_ZASTOSOWANIE | Roslina, RoslinaWlasna  |  V  Zastosowanie | Wskazuje zastosowanie rośliny. |  |

Tabela Relacje bez właściwości

###### Relacje z właściwościami

Relacja: ZASADZONA\_NA

Relacja ta wskazuje działkę, na której została zasadzona roślina. Połączenie (zob. rys. 14) kierunku *Roslina, RoslinaWlasna -> Dzialka*.

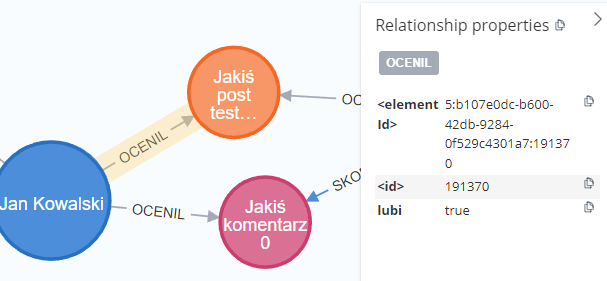
Właściwości relacji *ZASADZONA\_NA*:

* *x, y* – pozycja kafelka, którą zajmuje avatar(obraz) rośliny,
* *tabX, tabY* – dwie tablice jednowymiarowe kafelków (x, y) przechowujące pozycje kafelków zajmowanych przez roślinę – definiujące obszar zajmowany zajmowany przez roślinę na działce (Neo4j nie pozwala na użycie tablic wielowymiarowych)
* *kolor* – kolor kafelków rośliny,
* *obraz* – ścieżka do obrazu rośliny,
* *tekstura* – ścieżka do tekstury kafelków (jeśli użytkownik chce obrazować obszar zajmowany przez roślinę jako tekstura),
* *wyswietlanie* – tryb wyświetlania kafelków rośliny:
  + - KOLOR – pokolorowane kafelki,
    - TEKSTURA – tekstura kafelków (wymaga tekstury rośliny),
    - TEKSTURA\_KOLOR – pokolorowane kafelki z teksturą.



Rysunek Relacja ZASADZONA\_NA

Relacja: OCENIL

Relacja ta wskazuje węzeł oceniony przez użytkownika, którym jest oceniany post albo komentarz. Połączenie (zob. rys. 15) kierunku *Uzytkownik -> Post* albo *Uzytkownik -> Komentarz*.

Właściwości relacji *OCENIL*:

* *lubi* – ocena pozytywna (true) albo negatywna (false).

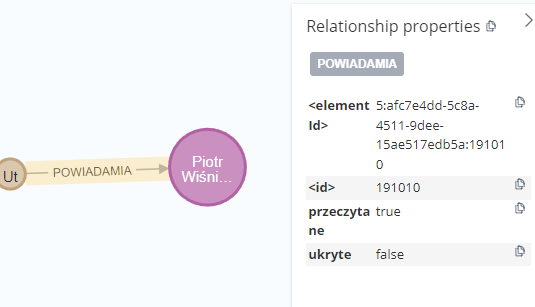
Rysunek Relacja OCENIL

Relacja: POWIADAMIA

Relacja ta wskazuje użytkownika, który jest powiadamiany. Połączenie (zob. rys. 16) kierunku *Powiadomienie -> Uzytkownik.*

Właściwości relacji *POWIADAMIA*:

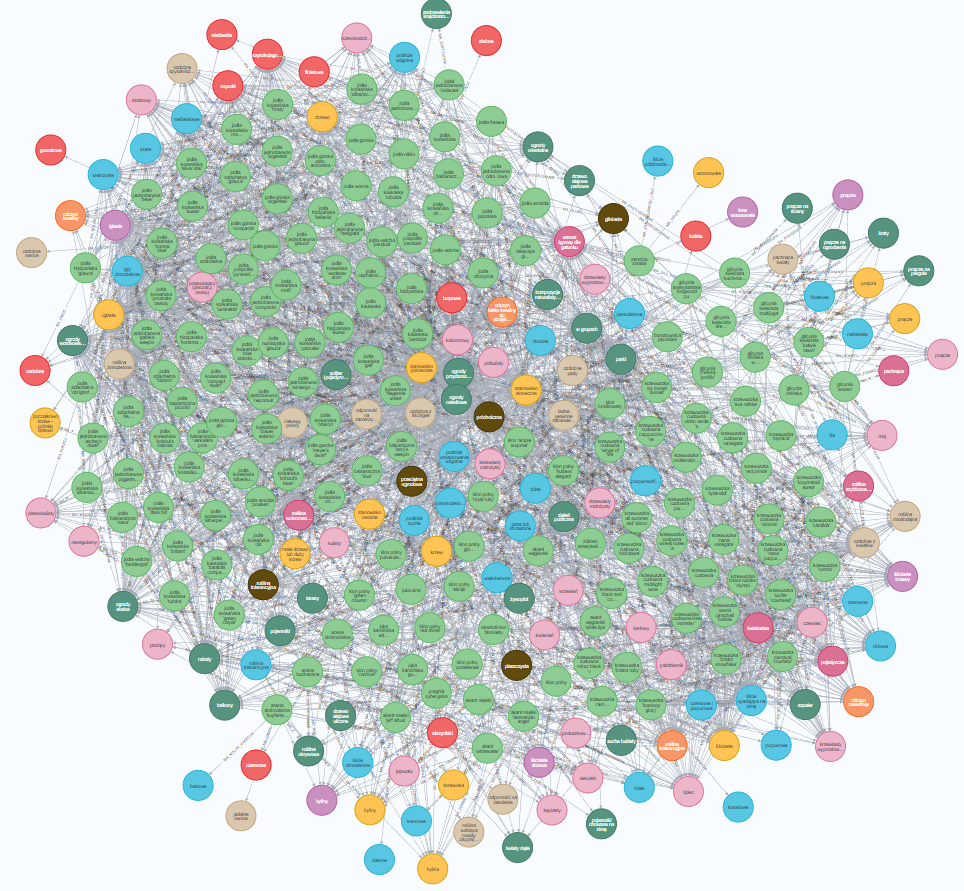
* *przeczytane* – status przeczytania powiadomienia (true/false),
* *ukryte* – status ukrycia powiadomienia przez użytkownika – dotyczy zgłoszeń i specjalnych powiadomień (true/false).



Rysunek Relacja POWIADAMIA

#### Widok bazy danych

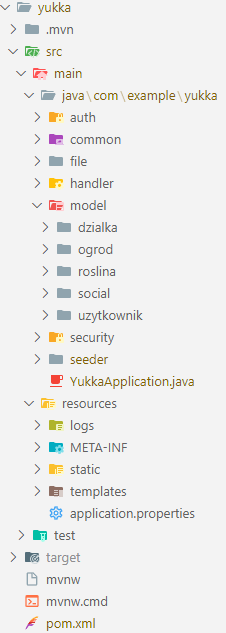
Utworzona w ramach pracy baza danych zawiera ponad 7 tysięcy węzłów i ponad 190 tysięcy relacji. Na rysunku 17 pokazano fragment grafu bazy danych składający się z 300 węzłów połączonych relacjami ze sobą. Ze względu na ograniczenia techniczne dotyczące renderowania dużej liczby węzłów w oprogramowaniu Neo4j, autor nie był w stanie pokazać widoku całej bazy.



Rysunek Widok bazy danych

### Struktura backendu aplikacji

Backend, oparty na frameworku Spring Boot [15], obsługuje żądania REST od frontendu, waliduje dane, wykonuje zapytania do bazy danych i zwraca odpowiedzi w formacie JSON [16].



Rysunek Struktura backendu

Projekt aplikacji backendowej (zob. rys. 18) podzielono na część programową zawierającą logikę aplikacji i część testową obejmującą testy jednostkowe weryfikujące działanie poszczególnych elementów. Poniżej scharakteryzowano poszczególne katalogi:

* *auth* – obsługa autoryzacji: logowanie, rejestracja, potwierdzenia e-mail, usuwanie konta,
* *common* – obiekty wspólne, np. odpowiedzi z paginacją,
* *file* – zarządzanie plikami i folderami,
* *handler* – obsługa błędów i odpowiedzi dla użytkownika,
* *model* – reprezentacja elementów bazy danych, kontrolery, serwisy, repozytoria, żądania i odpowiedzi:
  + *dzialka* – obsługa działek,
  + *ogrod* – obsługa ogrodów,
  + *roslina* – obsługa roślin,
  + *social* – obsługa społeczności (posty, komentarze, oceny, rozmowy, powiadomienia),
  + *uzytkownik* – obsługa użytkowników i ich usług.
* *security* – konfiguracja zabezpieczeń: uprawnienia, filtry, limity żądań,
* *seeder* – seedowanie bazy danych (przy seed-database=true),
* *YukkaApplication* – Główna klasa aplikacji,
* *resources* – logi, pliki .html (e-maile), *application.properties* (ustawienia aplikacji: porty, e-mail, JWT, ścieżki plików),
* *pom.xml* – plik z zależnościami programu.

#### Przykładowa klasa modelu danych

Klasa *Ogrod* reprezentuje węzeł *Ogrod* w bazie danych.  Posiada on właściwości odpowiadające reprezentowanemu węzłowi, a także zdefiniowane relacje o określonym typie i kierunku.

Adnotacje:

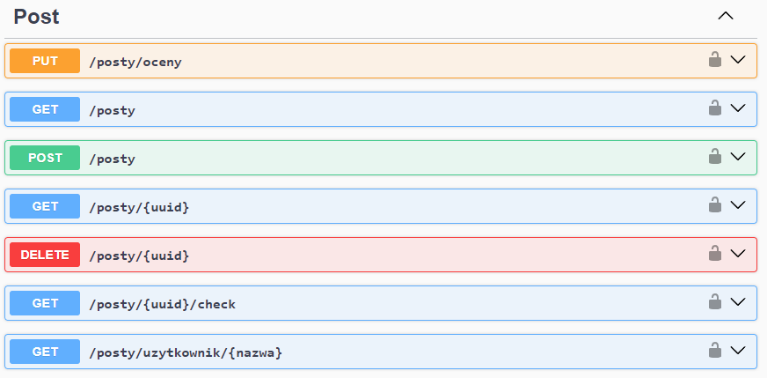
* *@Node* – definiuje węzeł w bazie danych,
* *@Property* – mapuje pola na właściwości węzła,
* *@Relationship* – definiuje typ i kierunek relacji,
* *@Getter, @Setter, @AllArgsConstructor, @NoArgsConstructor, @Builder* – automatyczne generowanie metod pomocniczych,
* *@Id, @GeneratedValue* – oznajmia, że jest to pole identyfikatora węzła, które jest generowane dla każdego obiektu mapowanego z bazy danych.



Rysunek Widok kodu źródłowego klasy Ogrod

#### Przykład procesu żądania i odpowiedzi

Aby użytkownik mógł wykonać operacje, musi on wysłać żądanie do serwera na backendzie. Na rysunku 20 znajduje się widok możliwych żądań dla Kontrolera o tagu[[4]](#footnote-4) „Post”:

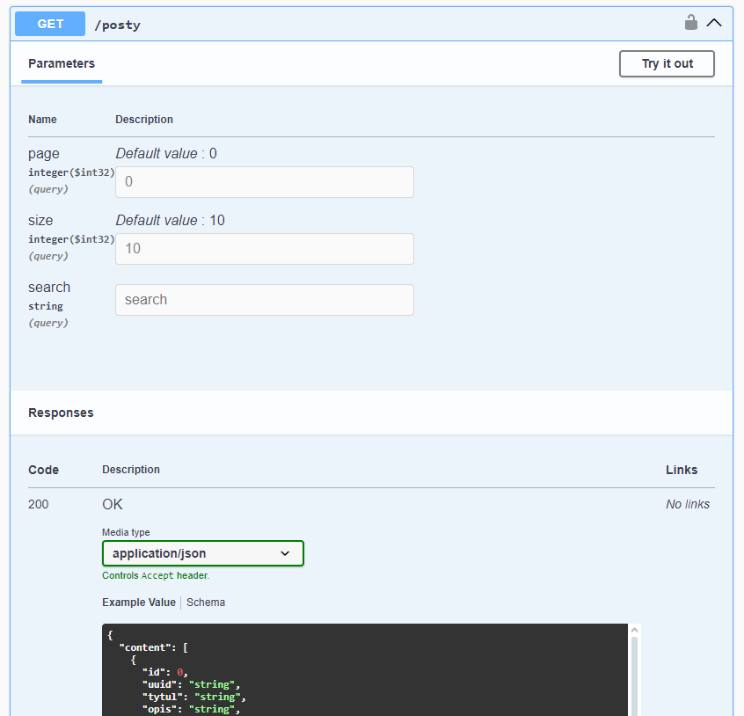


Rysunek Żądania dla kontrolera Post w Swaggerze

Kontroler posiada następujące ścieżki:

* *PUT /posty/oceny* – dodanie lub aktualizacja oceny postu,
* *GET /posty* – lista postów z uwzględnieniem paginacji,
* *POST /posty* – dodanie nowego postu,
* *GET /posty/{uuid}* – szczegóły postu o podanym identyfikatorze,
* *DELETE /posty/{uuid}* – usunięcie postu o podanym identyfikatorze,
* *GET /posty/{uuid}/check* – post o podanym identyfikatorze, bez komentarzy,
* *GET /posty/uzytkownik/{nazwa}* – lista postów użytkownika z uwzględnieniem paginacji.

Załóżmy, że użytkownik chce pobrać wszystkie posty. Do tego zadania należy użyć metody *GET /posty*. Na rysunku 21 znajduje się odpowiedni widok ze Swaggera [17]:



Rysunek Widok żądania GET /posty

Do żądania można opcjonalnie podać następujące parametry:

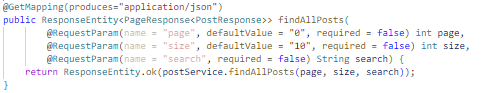
* *page* – strona o domyślnej wartości 0,
* *size* – liczba zwracanych elementów na stronę o domyślnej wartości 10,
* *search* – nazwa postu.

Przykładowa odpowiedź dla tego żądania zawiera listę postów wraz z takimi informacjami jak aktualna strona, całkowitą liczbę stron, całkowitą liczbę elementów, oraz czy jest to pierwsza, czy ostatnia strona (zob. rys. 22):



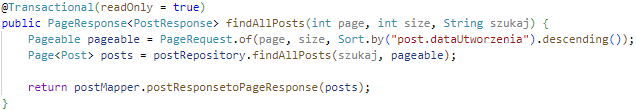
Rysunek Odpowiedź żądania

Z punktu widzenia kodu aplikacji żądanie wygląda w następujący sposób (zob. rys. 23):



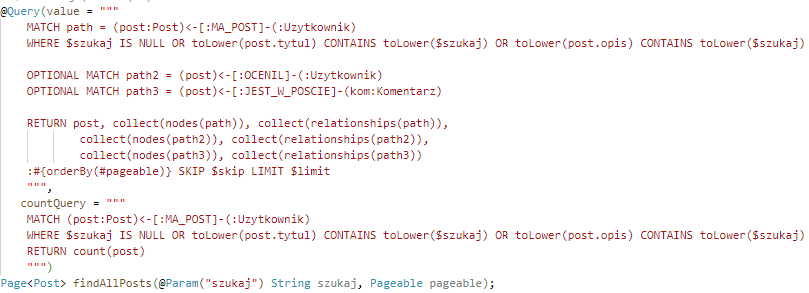
Rysunek Kod funkcji w kontrolerze

Kontroler *PostController* wykonuje funkcję *findAllPosts* z serwisu, której wynik opakowuje w ciało odpowiedzi o kodzie 200 (OK)[[5]](#footnote-5). Funkcję wykonywaną w opisywanym kontrolerze pokazano na rys. 24:



Rysunek Kod funkcji serwisu

W serwisie *PostService* funkcja *findAllPosts* używa funkcji *findAllPosts* z repozytorium *PostRepository*, zwracającą odpowiedź w postaci strony podanej w parametrze *page*. Ta odpowiedź jest następnie w funkcji serwisu mapowana odpowiednio do zwracanego wyniku. Funkcję wykonywaną w opisywanym repozytorium pokazano na rys. 25:



Rysunek Zapytanie do bazy danych

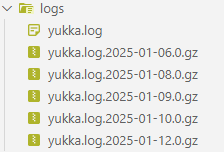
Funkcja ta wykonuje zapytanie w języku CYPHER, które przyjmuje parametr wyszukujący i obiekt mapujący wynik na stronę w paginacji, a następnie wysyła je do bazy danych, zwracając odpowiedź lub błąd. Zapytanie jest podzielone na dwa etapy. Pierwszy zwraca stronę wyników, a drugi całkowitą liczbę elementów. Po wykonaniu tych operacji, otrzymywana jest odpowiedź pokazana na rysunku 26. W tym przypadku zwraca ono listę postów, ich właściwości, oceny, liczbę komentarzy oraz dodatkowo nazwę i avatar (obraz) użytkownika, który stworzył post.



Rysunek Odpowiedź żądania pobrania postów

#### Logowanie zdarzeń i błędów

Aplikacja aktywnie loguje zdarzenia w aplikacji takie jak dodawanie nowych roślin, powiadomienia o błędach, czy też ostrzeżenia. Pliki logów są zapisywane w folderze *logs* (zob. rys. 27), gdzie przechowywane są logi wygenerowane w okresie ostatniego tygodnia:

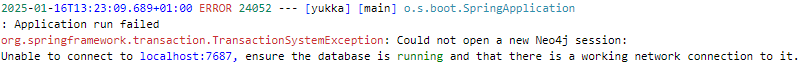


Rysunek Folder z logami

Poniżej pokazano przykładowy log: informacyjny, ostrzeżenia i błędu (zob. rys. 28):







Rysunek Przykład logów informacyjnego, ostrzeżenia i błędu

#### Zadania okresowe w tle

Zadania okresowe wykonywane w tle ograniczają się do sprawdzania zbanowanych użytkowników (codziennie) oraz sprawdzania okresów kwitnięcia i owocowania roślin na działkach (na początku miesiąca). Są one wykonywane dodatkowo przy starcie aplikacji. Przykładowo, na rysunku 29 funkcja *checkUnban()* odbanowuje użytkowników, dla których okres zablokowania się zakończył:

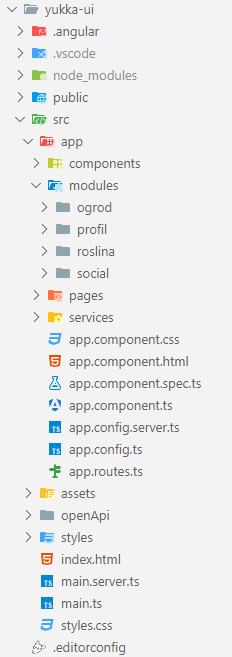


Rysunek Kod codziennego sprawdzania banów

### Struktura frontendu aplikacji

Część frontendowa aplikacji oparta jest na frameworku Angular. Część frontendowa obejmuje tworzenie interfejsu użytkownika, zawiera ona widoki aplikacji budowanych z pomocą komponentów komunikujących się między sobą głównie poprzez sygnały [18]. Struktura aplikacji (zob. rys. 30) zawiera tutaj widoki stron www, komponenty z których są zbudowane, serwisy wykonujące określone działania, a także moduły grupujące daną część aplikacji, zorganizowane w formie folderów. Poniżej opisano krótko najważniejsze foldery w strukturze aplikacji.

* *modules* – grupa powiązanych elementów, np. moduł profil zawiera komponenty, serwisy, routy[[6]](#footnote-6) i strony dla profilu użytkownika,
* *components* – niewielkie elementy aplikacji przeznaczone do wielokrotnego użytku, np. karta rośliny, karta postu, itp.,
* *services* – funkcje dla elementów, np. obsługa żądań do backendu
* *pages* – strony łączące moduły, serwisy i inne części aplikacji,
* *app.routes* – routy prowadzące do stron aplikacji.

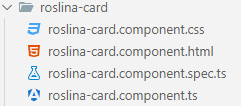


Rysunek Struktura frontendu

#### Przykładowy komponent

Struktura frontendu jest zbudowana z wielu komponentów. Przykładowo, komponent *roslina-card* (zob. rys. 31) służy do wyświetlania karty rośliny. Składa się z następujących plików:

* *roslina-card.component.css* – definicja styli dla komponentu,
* *roslina-card.component.html* – struktura widoku html komponentu,
* *roslina-card.component.spec.ts* – testy jednostkowe komponentu,
* *roslina-card.component.ts* –logika komponentu (dane, metody, zdarzenia).

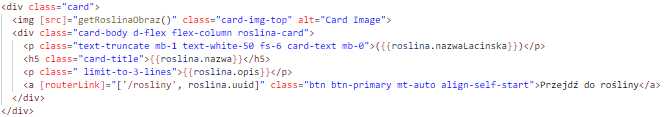
Rysunek Komponent roslina-card oraz jego widok

Plik *roslina-card.component.ts* (zob. rys. 32) zawiera importy potrzebnych modułów, klasę komponentu z obiektem wejściowym *@Input* dla rośliny, odpowiednie metody get i set oraz metodę zwracającą obraz rośliny w formacie base64[[7]](#footnote-7).



Rysunek Kod źródłowy komponentu roslina-card.component.ts

W pliku *roslina-card.component.html* (zob. rys. 33) wywoływana jest metoda *getRoslinaObraz()* zwracająca obraz rośliny do tagu *<img>*, a także wypisywane są właściwości rośliny.



Rysunek Kod źródłowy komponentu roslina-card.component.html

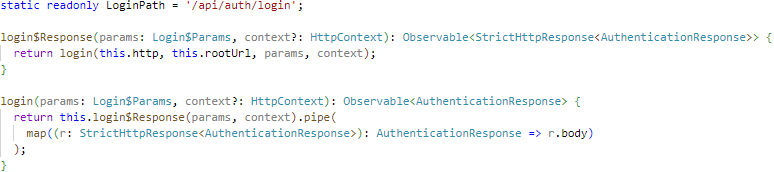
Komponent ten jest używany do pokazywania wyników z listy roślin, np. na rysunku 34 widać, że korzystając z pętli *ngFor*, można dla każdej rośliny w liście utworzyć komponent *roslina-card*, przekazując do niego dane konkretnej rośliny:



Rysunek Fragment kodu tworzący karty roślin

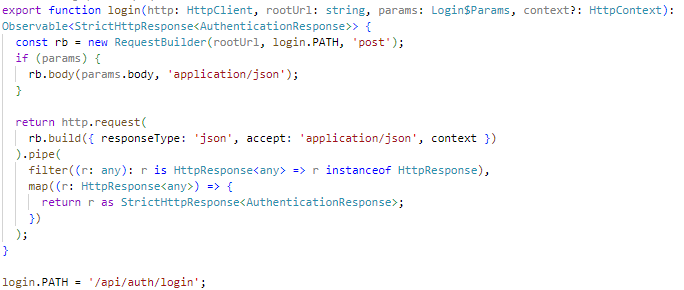
#### Przykład metody API z serwisu

Metody API zostały wygenerowane przy użyciu narzędzia *ng-openapi-gen*. Są podzielone na metody zwracające cały *HttpResponse*[[8]](#footnote-8) i metody zwracające samo ciało[[9]](#footnote-9). Każda metoda przyjmuje odpowiednie parametry oraz opcjonalnie HttpContext. Na rysunku 35 znajduje się przykładowa metoda API o nazwie *login* znajdująca się w pliku *authentication.service.ts:*



Rysunek Fragment kodu źródłowego pliku authentication.service.ts

Korzysta ona tutaj dodatkowo z wygenerowanej funkcji *login* [19] osadzonej w osobnym pliku *login.ts* (zob. rys. 36), która z kolei wykonuje żądanie logowania użytkownika zgodnie z dokumentacją OpenApi, przyjmując podane parametry i zwracając odpowiedź.



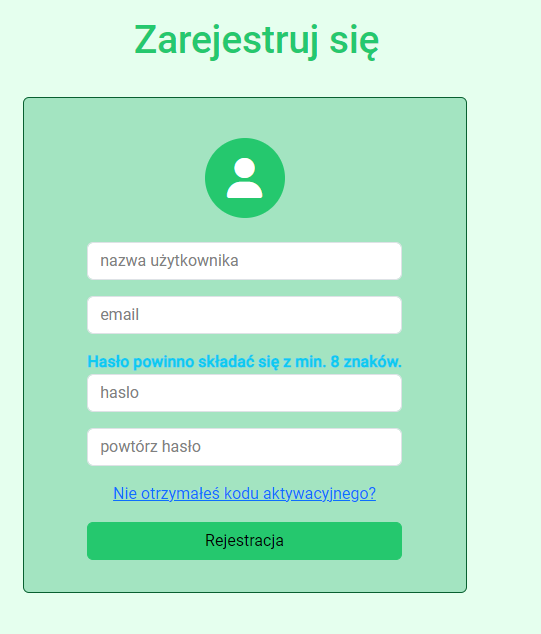
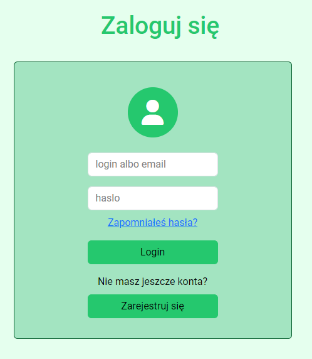
Rysunek funkcja login z pliku login.ts

## Opis funkcjonalności systemu

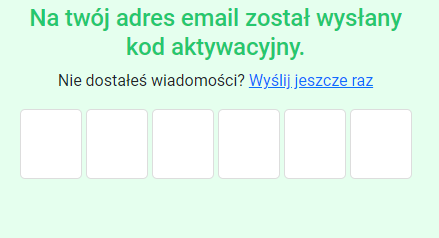
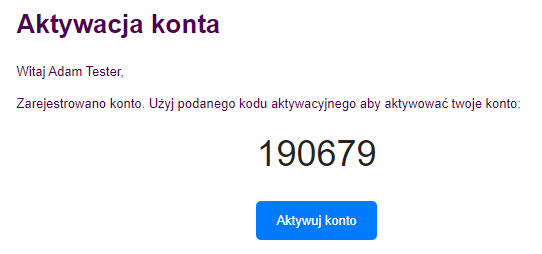
### Funkcjonalności zwykłego użytkownika

#### Rejestracja

W celu rejestracji w aplikacji, użytkownik klika w panelu nawigacyjnym przycisk „Zaloguj się”, który przenosi go do strony logowania (zob. rys. 37). Następnie w panelu wybrać opcję „Zarejestruj się” i wypełnić wymagane dane. Po rejestracji następuje przeniesienie na stronę aktywacji konta (zob. rys. 38), gdzie wprowadza się kod otrzymany na adres e-mail.



Rysunek Panel logowania i rejestracji

Rysunek Panel aktywacji i kod z wiadomości email

Aby się zalogować, należy w panelu logowania podać adres email lub nazwę użytkownika i hasło. Po kliknięciu przycisku „Login” użytkownik zostanie zalogowany i przeniesiony do strony głównej (zob. rys. 39).

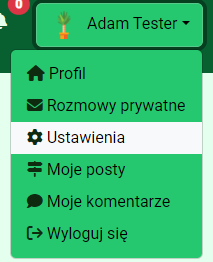
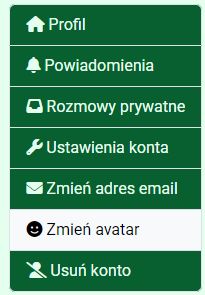


Rysunek Widok po zalogowaniu się

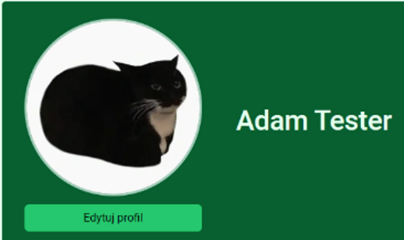
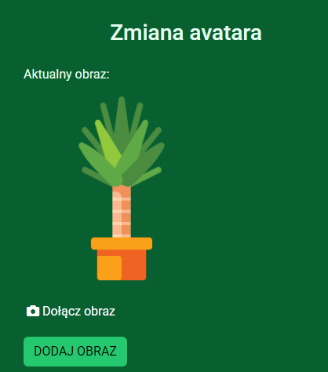
#### Zarządzanie kontem użytkownika

##### Zmiana avatara

Domyślnym avatarem dla nowych użytkowników jest ikonka roślinki pokazanej na rysunku 40. Aby użytkownik mógł ją zmienić, musi przejść do ustawień użytkownika i wybrać opcję „Zmień avatar”. Następnie w panelu (zob. rys. 41) dołączyć własny obraz, a następnie kliknąć „Dodaj obraz”.

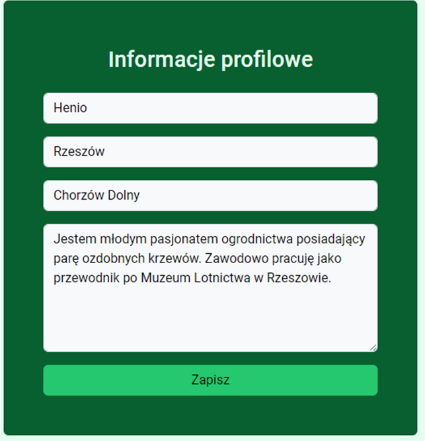
Rysunek Przejście do ustawień i do zmiany avatara



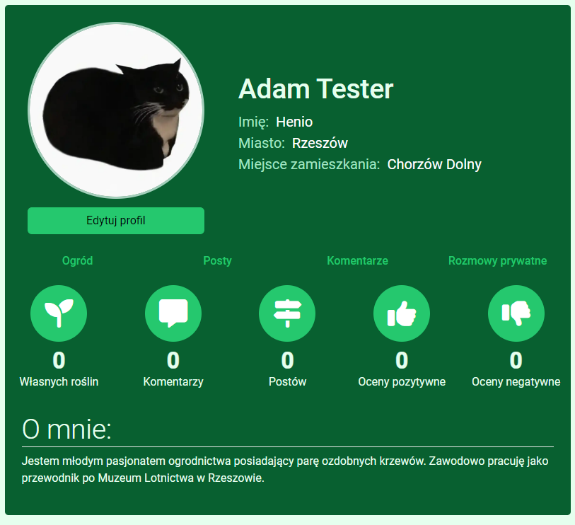
Rysunek Panel zmiany avatara i profil po zmianie

##### Edycja profilu

Aby użytkownik mógł zmienić dane profilowe, musi w panelu ustawień kliknąć „Profil” (zob. rys. 42). Można tutaj zmieniać lub usuwać imię, miasto, miejsce zamieszkania i opis. Po zapisaniu profil zostanie zmieniony (zob. rys. 43).



Rysunek Panel edycji profilu przed i po podaniu danych

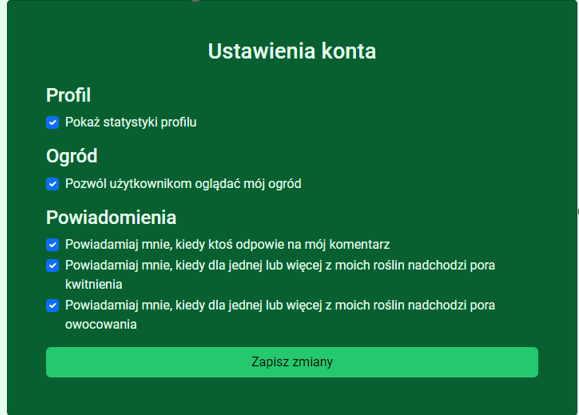


Rysunek Profil po edycji

##### Zmiana ustawień konta

Aby użytkownik mógł zmienić ustawienia konta, musi w panelu ustawień przejść do opcji „Ustawienia konta” (zob. rys. 44). W panelu można przełączyć:

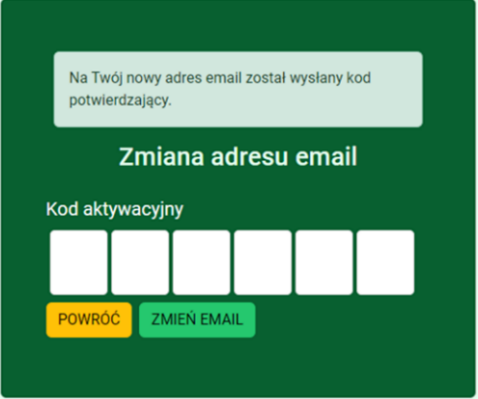
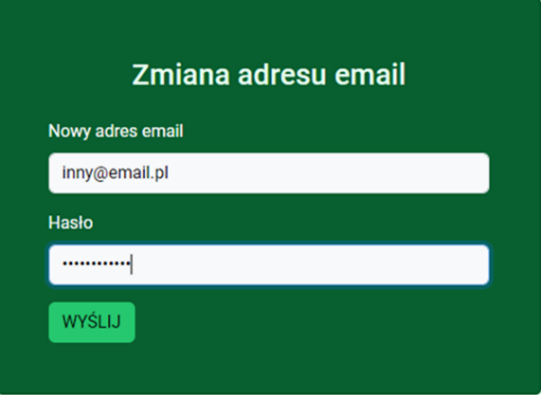
* pokazywanie statystyk profilu,
* ukrywanie ogrodu przed innymi użytkownikami,
* powiadomienia aplikacji.



Rysunek Panel ustawień konta

##### Zmiana adresu email

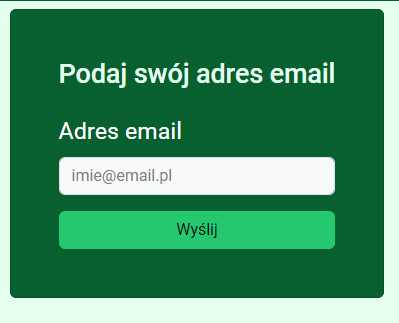
W celu zmiany adresu email, użytkownik musi w ustawieniach przejść do opcji „Zmień adres email”, gdzie trzeba w panelu (zob. rys. 45) podać nowy adres email oraz obecne hasło. Po kliknięciu przycisku „Wyślij” zostanie wysłana wiadomość z kodem na nowy adres email, który trzeba podać w następnym kroku. Po podaniu kodu i kliknięciu przycisku „Zmień email” pokaże się komunikat o zmianie adresu email.



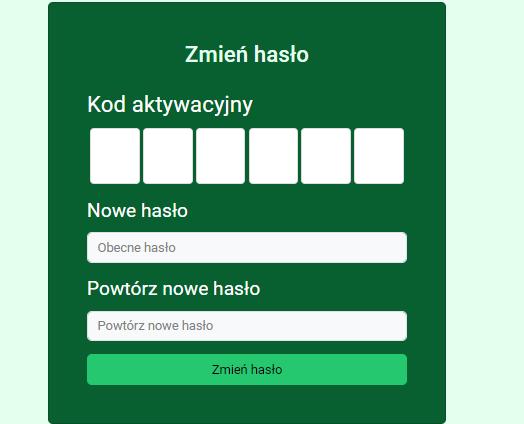
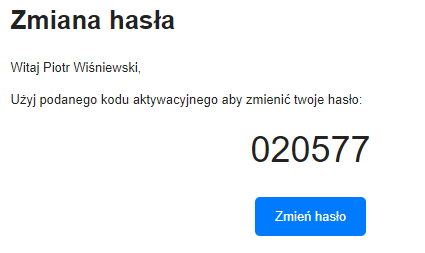
Rysunek Panel zmiany adresu email

##### Zmiana hasła

W celu zmiany hasła, użytkownik musi w panelu logowania kliknąć link „Zapomniałeś hasła?”, który przeniesie go do strony z resetowaniem hasła (zob. rys. 46). Po podaniu adresu email użytkownika, zostanie na ten adres wysłana wiadomość o zmianie hasła z kodem, po którego podaniu hasło zostanie zmienione (zob. rys. 47).



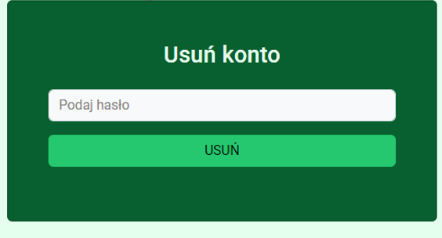
Rysunek Link do zmiany hasła i formularz



Rysunek Formularz zmiany hasła

##### Usuwanie konta

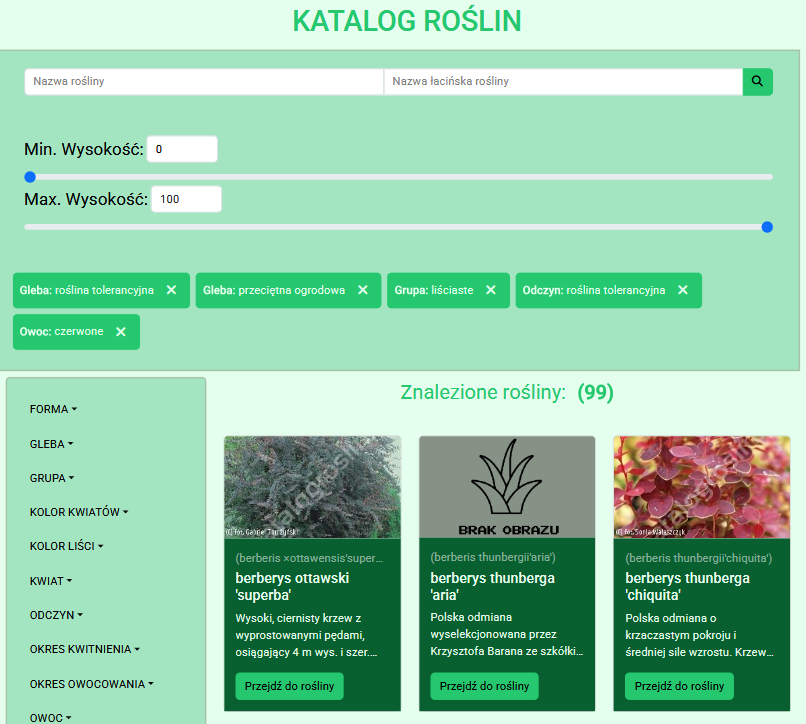
W celu usunięcia konta, użytkownik musi w ustawieniach wybrać panel „Usuń konto” (zob. rys. 48), gdzie po wpisaniu hasła i kliknięciu przycisku „Usuń” pojawi się ostrzeżenie, po którego potwierdzeniu użytkownik zostanie wylogowany i przeniesiony do strony głównej, a jego konto zostanie usunięte.



Rysunek Panel usuwania konta

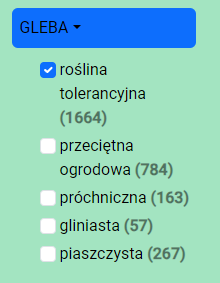
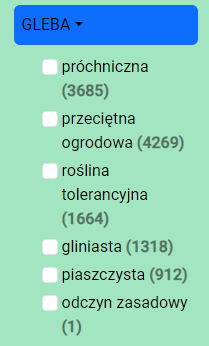
#### Przeglądanie roślin w katalogu

W celu przeglądania rośliny w katalogu, użytkownik musi wybrać „Przeglądaj katalog” w pasku nawigacyjnym, który przeniesie go na stronę katalogu roślin (zob. rys. 49). W katalogu można wyszukiwać rośliny według nazwy, nazwy łacińskiej, wysokości oraz cech (np. rodzaj gleby, miesiące kwitnienia), które są wprowadzane poprzez zaznaczenie wybranego elementu dla danego typu (zob. rys. 50). Przez strony katalogu można przechodzić, używając panelu paginacji na dole strony.





Rysunek Widok katalogu roślin



Rysunek Lista cech typu Gleba przed i po wyszukaniu

W celu zobaczenia rośliny z katalogu, użytkownik musi przejść do strony rośliny (zob. rys. 51), klikając na jej karcie przycisk „Przejdź do rośliny”.



Rysunek Widok strony rośliny

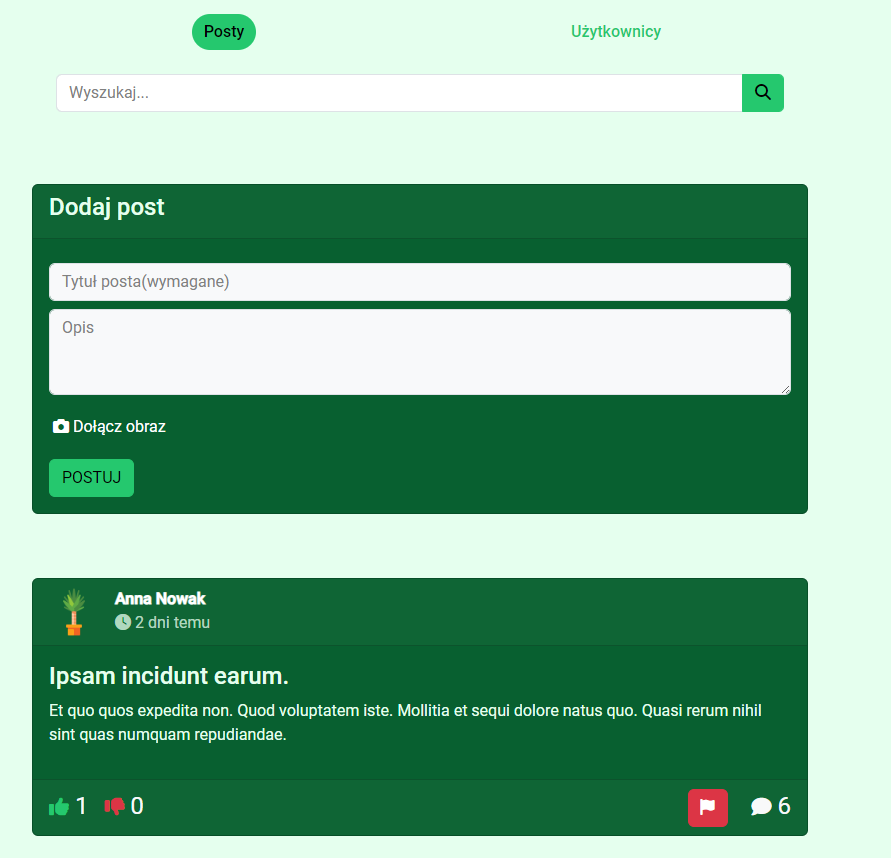
#### Zarządzanie własnymi roślinami

W celu zarządzania własnymi roślinami, użytkownik musi w pasku nawigacyjnym wybrać opcję „Mój ogród”, a następnie w części „Rośliny użytkownika” kliknąć przycisk „Zobacz”, który przeniesie użytkownika do jego katalogu z roślinami, który funkcjonuje dokładnie jak w rozdziale 5.1.3 Przeglądanie roślin w katalogu z tą różnicą, że przegląda on własne rośliny, którymi może zarządzać.

#### Zarządzanie postami i komentarzami

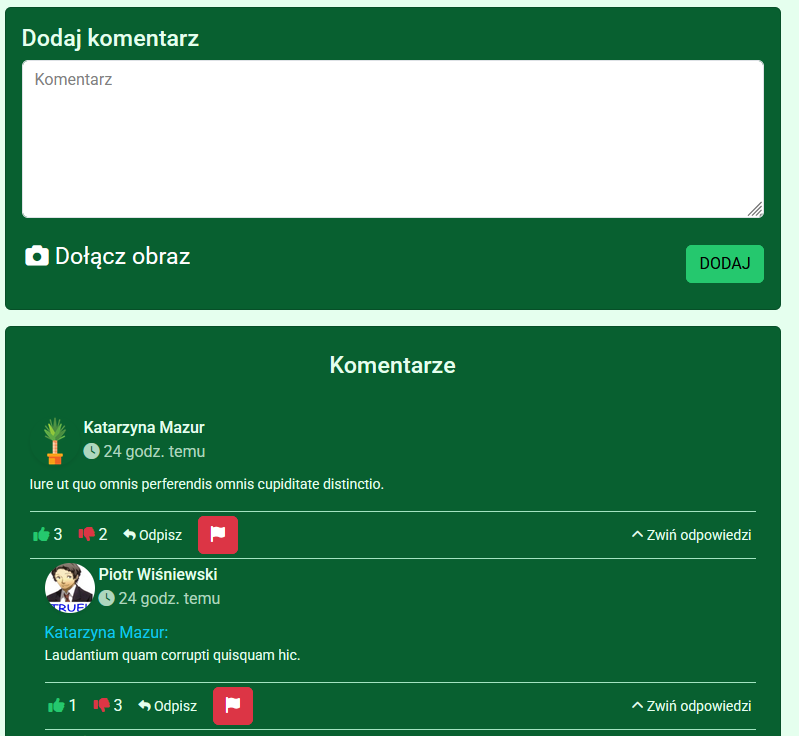
##### Przeglądanie postów i komentarzy

W celu przeglądania postów, użytkownik musi w panelu nawigacyjnym wybrać opcję „Społeczność”, który przeniesie go do panelu społeczności pokazanego na rys. 52. Panel społeczności posiada wyszukiwarkę postów i użytkowników. W postach zastosowano mechanizm „Infinite Scrolling”, ładujący nowe posty po dotarciu do końca strony [20]. Same posty wyświetlane są w postaci kart pokazujących autora z jego avatarem, tytuł postu, opis, obraz (opcjonalnie), oceny oraz liczbę jego komentarzy.



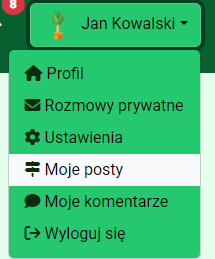
Rysunek Wyszukiwarka postów , panel dodawania postu i przykładowy post

W celu przejścia do strony postu, użytkownik musi kliknąć na liczbę komentarzy, która przeniesie go do strony postu (zob. rys. 53). Na stronie postu oprócz wspomnianego postu znajduje się lista wszystkich komentarzy oraz możliwość dodania komentarza. Komentarze mają opcje oceniania, odpowiadania i zwijania zagnieżdżonych odpowiedzi.



Rysunek Widok komentarzy postu

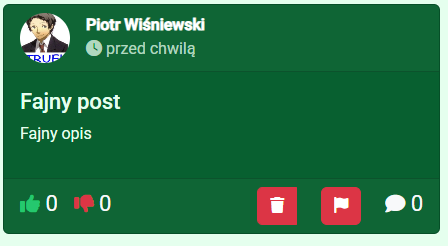
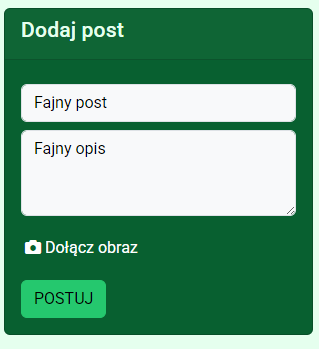
Użytkownik może przeglądać własne posty i komentarze, klikając w panelu użytkownika (zob. rys. 54) opcję „Moje posty” lub „Moje komentarze”, które przekieruje użytkownika do strony z jego postami czy komentarzami:



Rysunek Przejście do postów użytkownika

##### Dodanie postu

W celu dodania postu, użytkownik musi w panelu „Społeczność”, wypełnić formularz (zob. rys. 55), podając tytuł oraz (opcjonalnie) opis i obraz postu. Po zatwierdzeniu użytkownik zostanie przeniesiony do strony postu. Użytkownik może dodawać nowy post co 30 sekund.



Rysunek Formularz dodania postu i post po dodaniu

##### Usuwanie postu

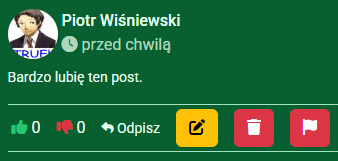
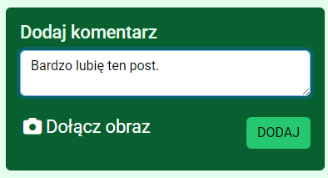
W celu usunięcia postu, użytkownik musi kliknąć ikonę śmietnika i zatwierdzić (zob. rys. 56), po którym post zostanie usunięty, a użytkownik zostanie przekierowany z powrotem do strony społeczności.



Rysunek Usuwanie postu

##### Dodanie komentarza

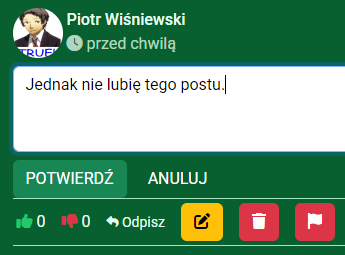
W celu dodania komentarza do postu, użytkownik musi na stronie postu wypełnić formularz (zob. rys. 57), po którego zatwierdzeniu komentarz zostanie dodany. Użytkownik może dodawać nowy komentarz co 30 sekund.



Rysunek Dodawanie komentarza i dodany komentarz

##### Edycja komentarza

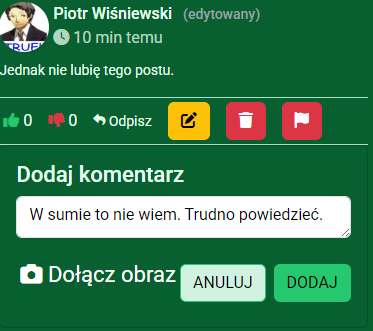
W celu edycji komentarza, użytkownik musi kliknąć żółtą ikonę edycji wywołującą formularz, po którego wypełnieniu komentarz zostanie zmieniony (zob. rys. 58):



Rysunek Edycja komentarza i komentarz po edycji

##### Dodanie odpowiedzi

Aby dodać odpowiedź do komentarza, należy kliknąć „Odpisz” w komentarzu, a następnie wypełnić formularz (zob. rys. 59):



Rysunek Dodawanie odpowiedzi i odpowiedź po edycji

##### Ocena postu lub komentarza

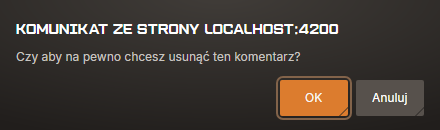
Użytkownik może ocenić post lub komentarz, klikając na kciuk w górę lub w dół (zob. rys. 60):



Rysunek Ocena postu/komentarza

##### Usuwanie komentarza

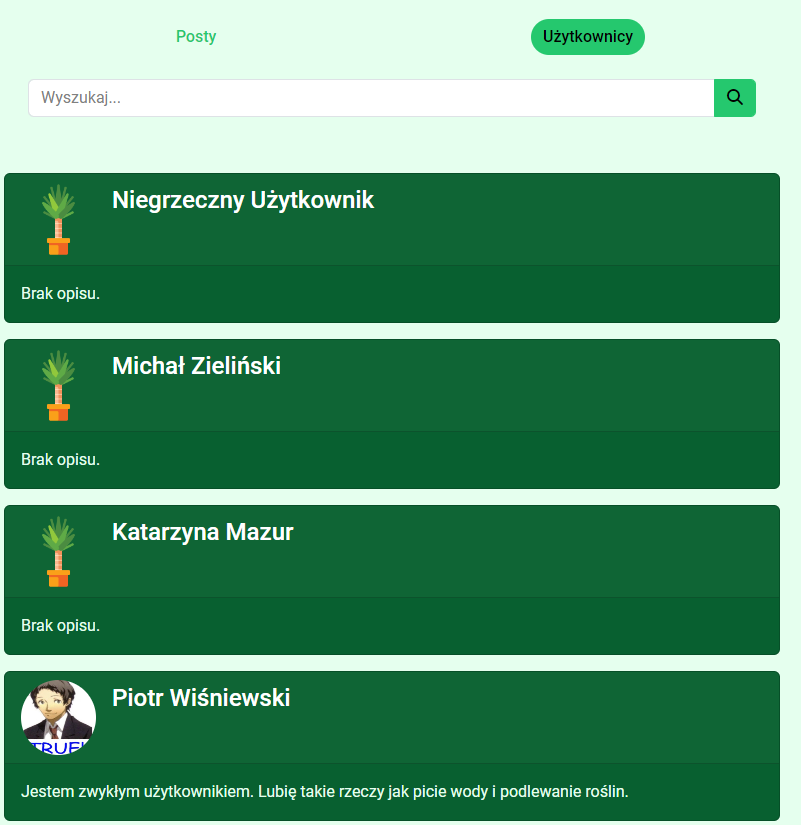
W celu usunięcia komentarza, użytkownik musi kliknąć ikonę śmietnika i potwierdzić (zob. rys. 61).



Rysunek Ikona usuwania

#### Przeglądanie użytkowników

W celu przeglądania użytkowników, użytkownik musi w panelu „Społeczność” kliknąć na przycisk „Użytkownicy”, który zabierze użytkownika do listy użytkowników (zob. rys. 62):



Rysunek Lista użytkowników i profil jednego z nich

#### Zarządzanie ogrodem

##### Przeglądanie ogrodu

W celu przeglądania ogrodu, użytkownik musi przejść do ogrodu klikając opcję „Mój ogród” w pasku nawigacyjnym lub wybierając opcję „Ogród” na profilu innego użytkownika. Na stronie ogrodu (zob. rys. 63) wyświetlana jest jego nazwa, lista działek, oraz przenoszący użytkownika do katalogu roślin stworzonych przez danego użytkownika.

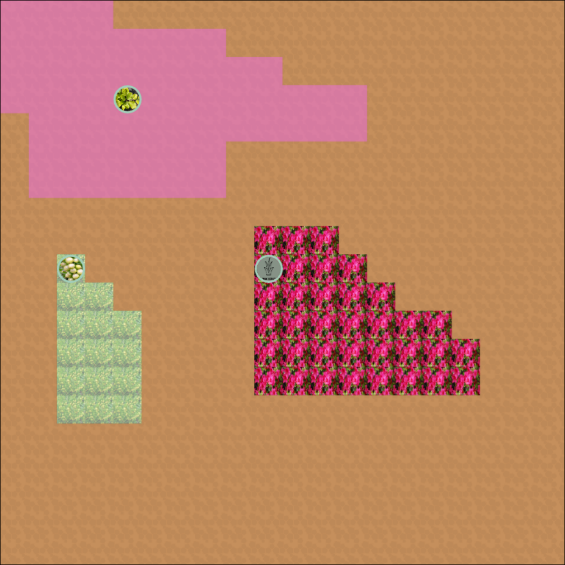


Rysunek Widok ogrodu użytkownika

Do strony działki można przejść, klikając przycisk „Zobacz” na karcie działki. Na stronie działki (zob. rys. 64) znajduje się nazwa działki i rośliny pokazanych w formie obrazów i kafelków. Każda dodana roślina posiada wybrany tryb wyświetlania. Użytkownik może przybliżać/oddalać widok działki oraz przemieszczać się po niej poprzez przeciąganie. Dodatkowo, można zapisać obraz działki klikając w ikonę dyskietki i potwierdzając (zob. rys. 65).



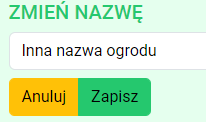
Rysunek Widok działki



Rysunek Stworzony obraz działki

##### Zmiana nazwy ogrodu lub działki

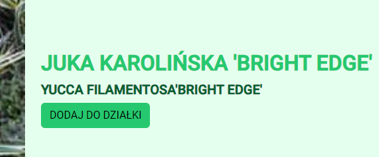
W celu zmiany nazwy ogrodu lub działki, użytkownik musi kliknąć ikonę po prawej od nazwy, a następnie w polu wprowadzić nową nazwę i potwierdzić (zob. rys. 66):

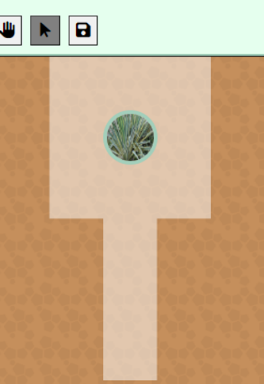
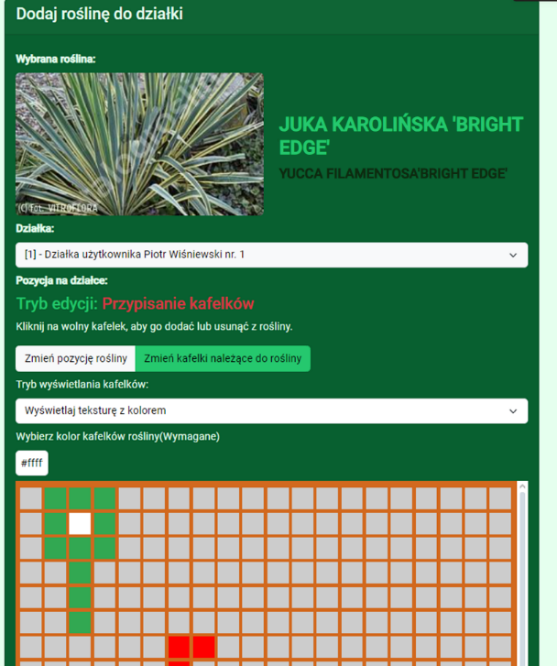
Rysunek Zmiana nazwy ogrodu

##### Dodanie rośliny do działki

W celu dodania rośliny do działki, użytkownik musi przejść do strony rośliny i kliknąć przycisk „Dodaj do działki” (zob. rys. 67), który przeniesie go do strony dodawania rośliny do działki (zob. rys. 68).



Rysunek Przycisk "Dodaj do działki"



Rysunek Strona dodawania rośliny do działki i nowo dodana roślina

Przed dodaniem rośliny do działki, użytkownik musi wykonać następujące czynności:

1. wybrać działkę,
2. wybrać tryb wyświetlania kafelków:

* wyświetlanie samego koloru,
* wyświetlanie samej tekstury (tylko, jeśli podano obraz tekstury),
* wyświetlanie tekstury z nałożonym kolorem.

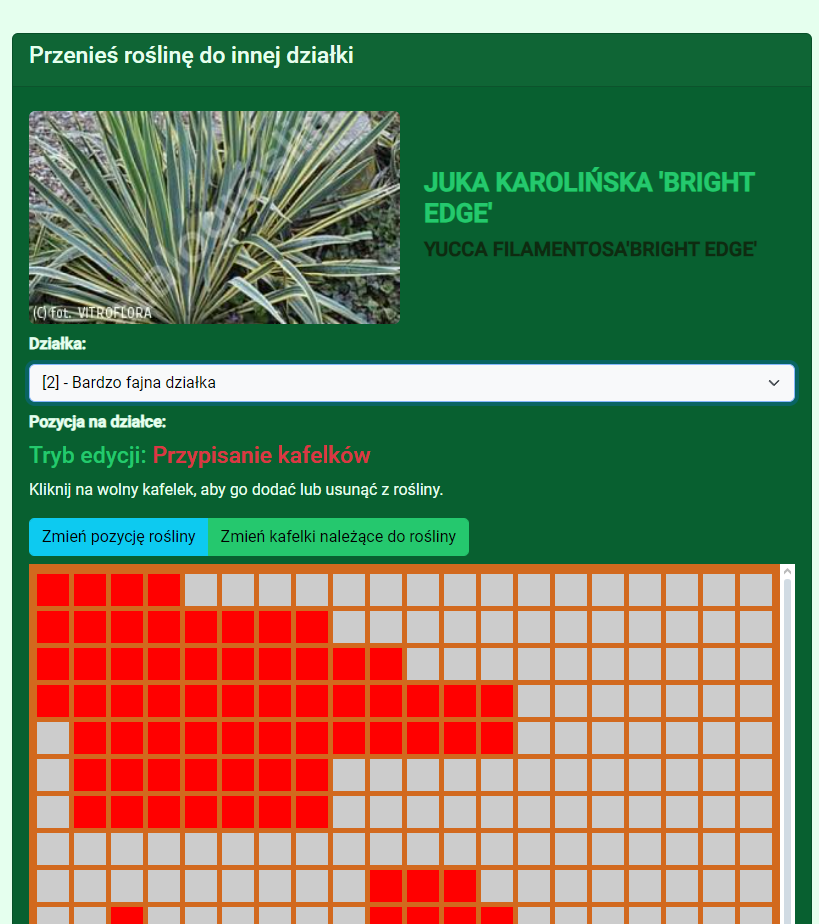
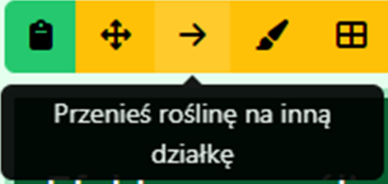
1. wybrać kafelki, które zajmie roślina:

* *białe:* reprezentuje pozycję avatara rośliny,
* *zielone:* reprezentuje zaznaczone kafelki rośliny,
* *szare:* reprezentuje wolne kafelki,
* *czerwone:* reprezentuje kafelki zajęte przez inne rośliny.

1. (Opcjonalnie) wybrać obraz rośliny,
2. (Opcjonalnie) wybrać teksturę kafelków,
3. wcisnąć przycisk „Dodaj roślinę”, po którym zostanie się przeniesionym do działki z nowo dodaną rośliną.

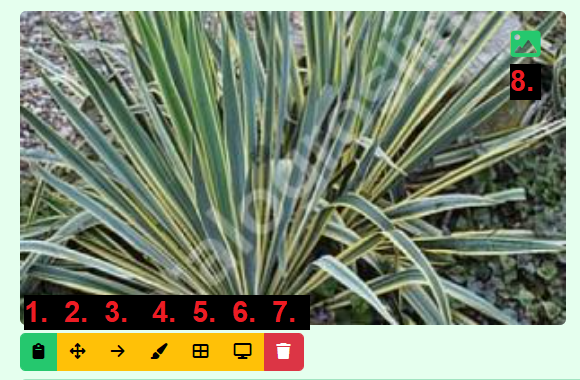
##### Przeniesienie rośliny do innej działki

W celu przeniesienia rośliny do innej działki, użytkownik musi w trybie wybierania kliknąć na roślinę, a następnie kliknąć przycisk przenoszenia do innej działki, który zaprowadzi go do strony przenoszenia (zob. rys. 69). Następujące czynności są takie same, jak w poprzednim rozdziale Dodanie rośliny do działki z tym wyjątkiem, że ustawia się jedynie pozycję rośliny na innej działce.



Rysunek Przycisk przenoszenia do innej działki i widok

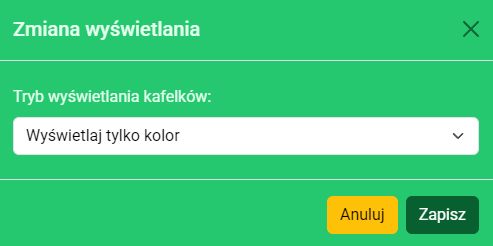
##### Zmiana właściwości rośliny

Użytkownik może zmieniać właściwości rośliny na działce, przechodząc do jej panelu (zob. rys. 70), gdzie każdy przycisk może zmienić jej właściwość:

1. Dodanie notatki
2. Zmiana pozycji na działce
3. Przeniesienie do innej działki
4. Zmiana koloru kafelków
5. Zmiana tekstury kafelków
6. Zmiana trybu wyświetlania
7. Usunięcie rośliny z działki
8. Zmiana obrazu

Rysunek Opcje zmiany właściwości rośliny

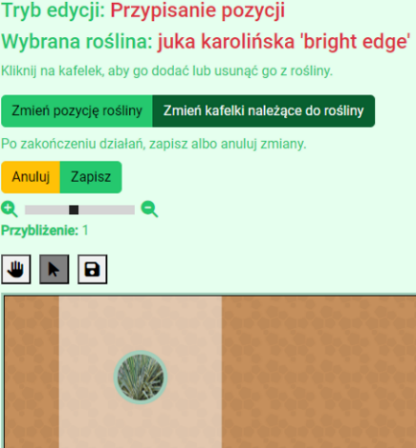
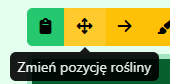
Przykładowo, użytkownik może zmienić tryb wyświetlania kafelków rośliny, klikając na ikonę nr.6. Pojawi się wtedy okienko (zob. rys. 71), gdzie wybiera się jeden z trybów wyświetlania, który zostaje dynamicznie zmieniony na działce po zatwierdzeniu:



Rysunek Okno z wybieraniem koloru i roślina po zmianie

##### Zmiana pozycji rośliny na działce

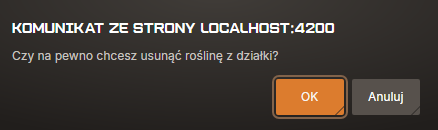
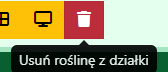
W celu zmiany pozycji rośliny na działce, użytkownik musi w trybie wybierania wybrać roślinę, wybierając w jej panelu opcję „Zmień pozycję rośliny” (zob. rys. 72). Podobnie jak w dodawaniu rośliny do działki, użytkownik musi wybrać kafelki i pozycję rośliny. Po zatwierdzeniu zmian przyciskiem „Zapisz”, roślina na działce zostanie zaktualizowana:



Rysunek Zmiana pozycji rośliny

##### Usuwanie rośliny z działki

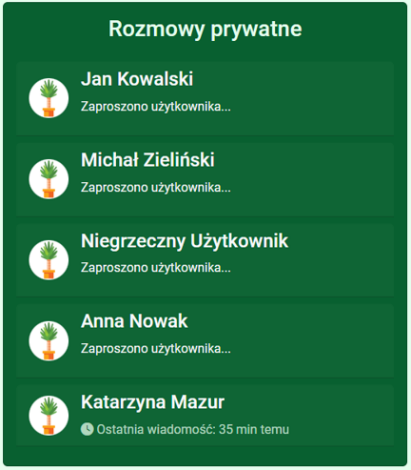
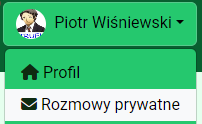
W celu usunięcia rośliny z działki, użytkownik musi kliknąć ikonę śmietnika i potwierdzić (zob. rys. 73).



Rysunek Usuwanie rośliny z działki

#### Zarządzanie rozmowami

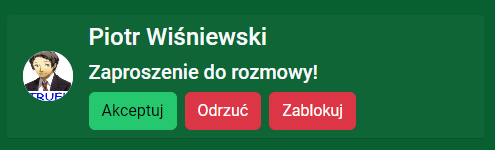
W celu przeglądania rozmów, użytkownik musi przejść do widoku rozmów wybierając w swoim panelu opcję „Rozmowy prywatne”. W panelu rozmów prywatnych znajdują się rozmowy zalogowanego użytkownika z innymi, a także oczekujące zaproszenia wysłane lub odebrane.



Rysunek Przejście do rozmów i rozmowy prywatne

##### Wysyłanie zaproszenia

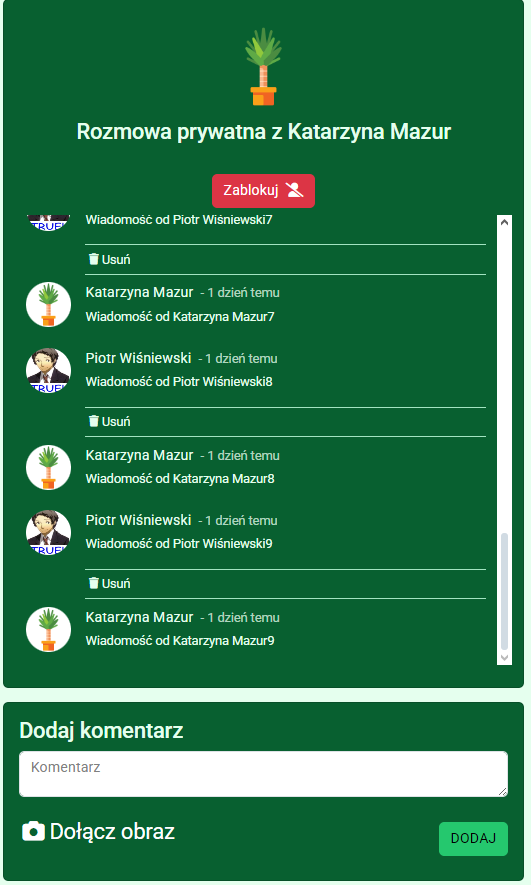
W celu wysłania zaproszenia, użytkownik musi na profilu innego użytkownika kliknąć przycisk „Zaproś”. Po wysłaniu zaproszenia, odbiorca może je zaakceptować, odrzucić albo zamiast tego zablokować nadawcę (zob. rys. 75). Akceptacja zaproszenia umożliwia obu użytkownikom wysyłanie wiadomości między sobą.



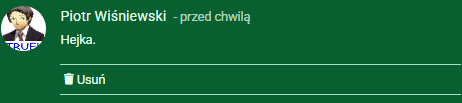
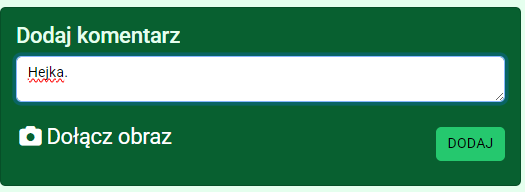
Rysunek Przycisk "Zaproś" i wysłane zaproszenie

##### Wysyłanie wiadomości

W celu wysłania wiadomości do aktywnej rozmowy, użytkownik musi w liście rozmów (zob. rys. 76) przejść do wybranej rozmowy, klikając na jej kartę. Na stronie rozmowy musi on wtedy podać takie same dane, jak do komentarza w rozdziale 5.1.5 Zarządzanie postami i komentarzami.



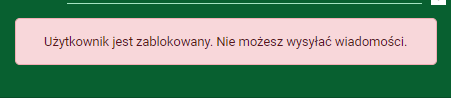
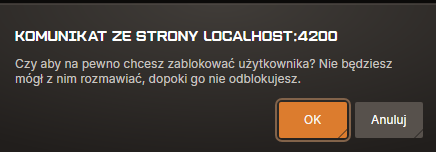
Rysunek Widok rozmowy prywatnej



Rysunek Dodanie wiadomości

##### Blokowanie użytkownika

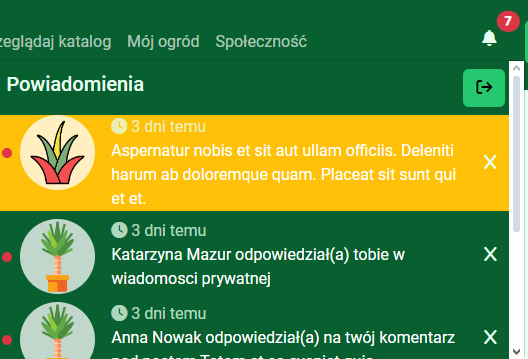
W celu zablokowania wybranego użytkownika w rozmowie, użytkownik musi na profilu wybranego użytkownika, lub w panelu rozmowy kliknąć przycisk „Zablokuj”, a następnie potwierdzić operację (zob. rys. 78). Po zablokowaniu użytkownika nie jest możliwe wysyłanie do niego wiadomości, ocenianie ani komentowanie jego postów oraz komentarzy.



Rysunek Potwierdzenie blokowania i wiadomość po zablokowaniu

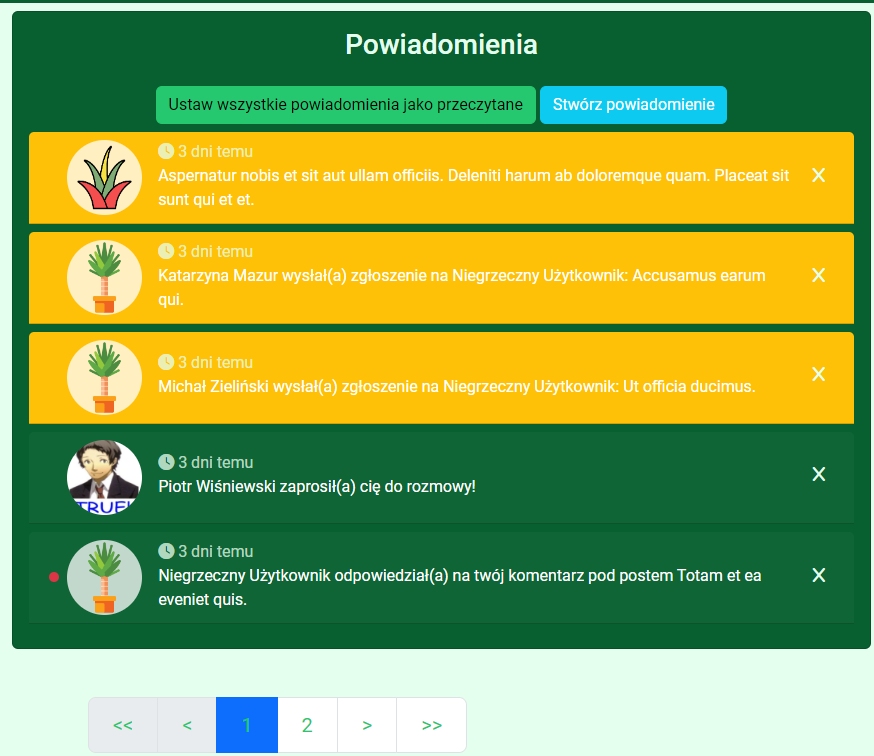
#### Zarządzanie powiadomieniami

W celu zarządzania powiadomieniami, użytkownik musi na pasku nawigacyjnym kliknąć w ikonę dzwonka, który pokazuje listę powiadomień (zob. rys. 79). Powiadomienia oznaczane są jako przeczytane po najechaniu na nie myszką, a także można je usuwać przyciskiem „X”. Zgłoszenia i powiadomienia specjalne są oznaczone żółtym kolorem.



Rysunek Okno powiadomień

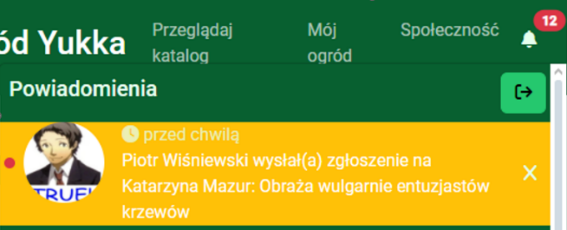
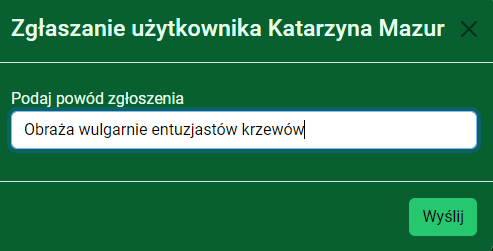
Przycisk w prawym górnym rogu przenosi użytkownika do strony z powiadomieniami (zob. rys. 80) ukazywanych w formie paginacji. Na stronie oprócz wspomnianych wcześniej funkcjonalności, użytkownik może oznaczyć wszystkie powiadomienia jako przeczytane lub utworzyć powiadomienie specjalne (dostępne tylko dla pracowników i administratorów).



Rysunek Panel powiadomień pracownika

##### Dodawanie zgłoszeń

W celu dodania zgłoszenia, użytkownik musi kliknąć ikonę flagi na profilu innego użytkownika, jego poście lub komentarzu. Po kliknięciu wyskoczy okno z formularzem zgłaszania, po którego wypełnieniu zgłoszenie zostanie wysłane do jednego z pracowników (zob. rys. 81).

Rysunek Opis zgłoszenia i widok wysłanego zgłoszenia

##### Usuwanie powiadomienia

W celu usunięcia powiadomienia, użytkownik musi kliknąć „X” po prawej stronie powiadomienia. Operacja ta usuwa relację użytkownika z powiadomieniem, ale samo powiadomienie jest trwale usuwane dopiero wtedy, gdy żaden użytkownik nie ma z nim relacji.

Uwaga: Zgłoszenia nie są usuwane, lecz ukrywane, ponieważ są przypisane do pracowników z najmniejszą liczbą zgłoszeń.

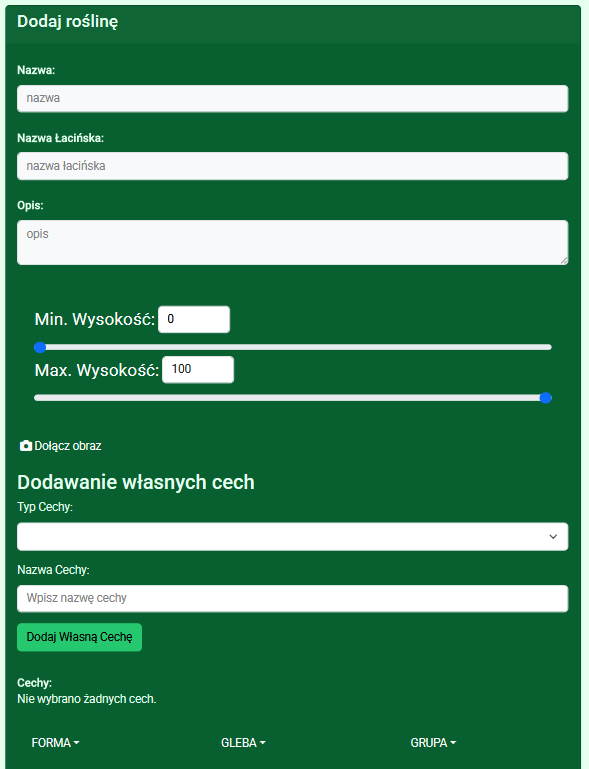
### Funkcjonalności pracownika

Pracownik dziedziczy wszystkie funkcjonalności użytkownika. Oprócz tego posiada on swoje własne zdefiniowane w rozdziale 3.2 Określenie funkcjonalności aplikacji.

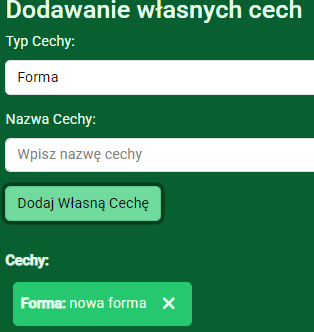
#### Zarządzanie roślinami

##### Dodawanie rośliny

W celu dodania rośliny do katalogu, użytkownik musi w katalogu roślin kliknąć przycisk „Dodaj roślinę”, który przeniesie go do strony dodawania (zob. rys. 82). Jeśli zamiast tego chce użytkownik pragnie dodać własną roślinę, musi on przejść do strony „Mój ogród”, i tam kliknąć przycisk „Dodaj roślinę”. Przed dodaniem rośliny, użytkownik musi wpisać parametry rośliny, opcjonalnie dodać obraz i przypisać wybrane rodzaje cech. Możliwe jest również stworzenie jednej z cech poprzez wybranie typu i podanie własnej nazwy (zob. rys. 83).



Rysunek Przycisk dodawania rośliny i fragment formularza



Rysunek Nowa cecha rośliny

Po podaniu danych i zatwierdzeniu, użytkownik zostanie przeniesiony do strony nowo dodanej rośliny (zob. rys. 84).

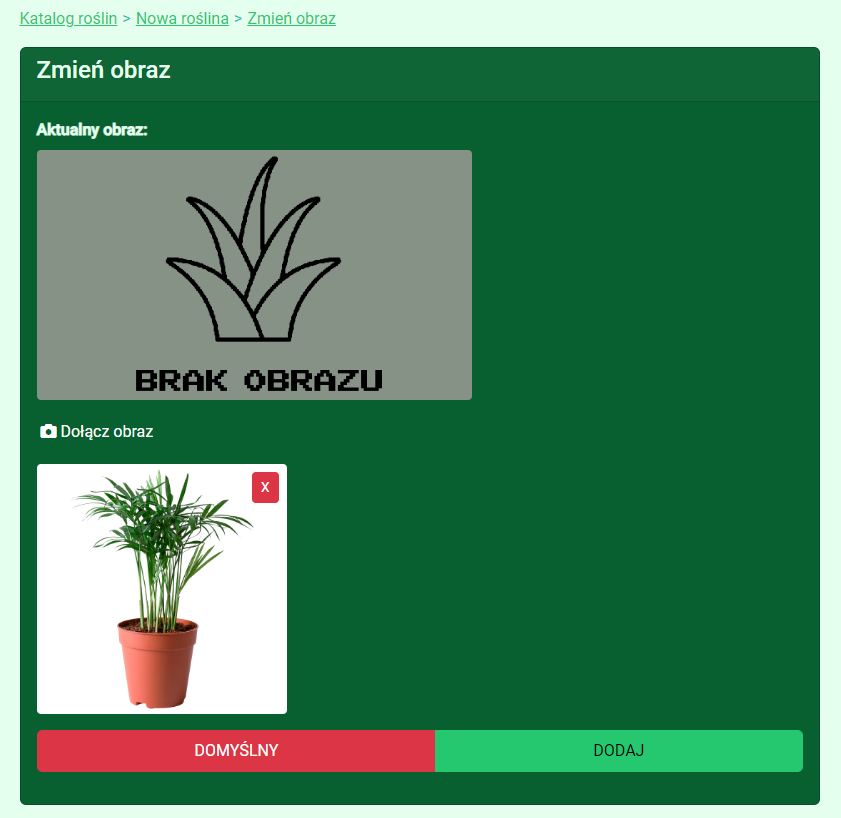


Rysunek Dodana roślina

##### Aktualizacja rośliny

W celu aktualizacji rośliny, użytkownik musi na jej stronie kliknąć przycisk „Edytuj”, który przeniesie go do strony edycji rośliny. Jest ona podobna do strony dodawania rośliny z tą różnicą, że nie ma opcji dodawania obrazu. Po zatwierdzeniu zmian zostanie się z powrotem przeniesionym do strony rośliny.

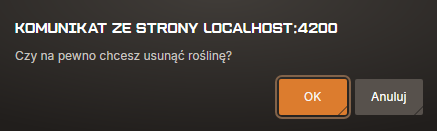
Jeśli użytkownik pragnie zmienić obraz rośliny, musi kliknąć przycisk „Zmień obraz” na stronie rośliny, który przeniesie go do strony zmiany obrazu (zob. rys. 85), gdzie może wstawić nowy obraz lub przywrócić domyślny. Po dokonaniu wyboru obraz rośliny zostanie zaktualizowany.

Rysunek Zmiana obrazu rośliny

##### Usuwanie rośliny

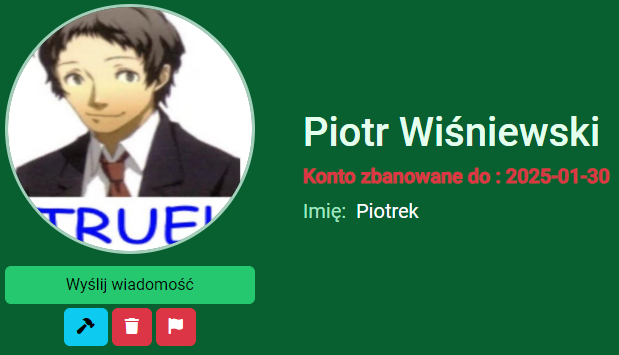
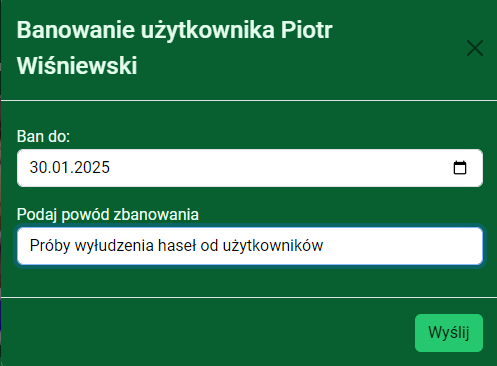
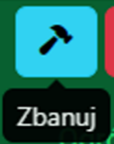
W celu usunięcia rośliny, użytkownik musi na jej stronie kliknąć przycisk „usuń” i zatwierdzić (zob. rys. 86). Po usunięciu rośliny uzytkownik zostanie z powrotem przeniesiony do katalogu.



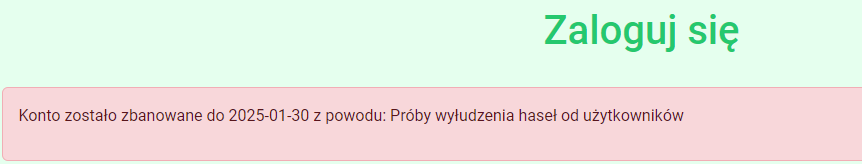
Rysunek Potwierdzenie usuwania rośliny

#### Banowanie użytkowników

W celu zbanowania użytkownika, pracownik musi przejść do profilu wybranego użytkownika i kliknąć ikonę młotka, która wywołuje okno banowania, gdzie trzeba podać datę zakończenia bana oraz powód (zob. rys. 87). Po zatwierdzeniu konto zostanie zbanowane aż do wybranej daty. Możliwe jest również ręczne odbanowanie użytkownika również klikając na ikonę młotka i potwierdzając.



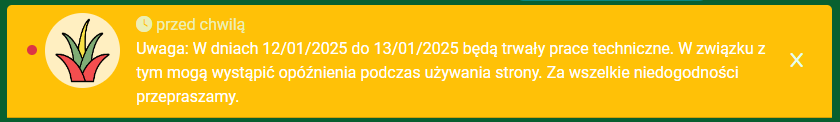
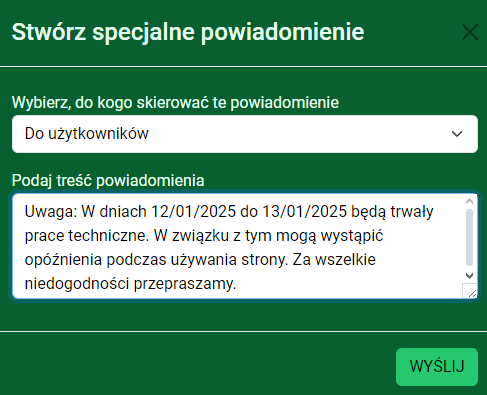
Rysunek Formularz banowania i użytkownik po zbanowaniu



Rysunek Komunikat o zbanowaniu przy próbie zalogowania się

#### Tworzenie specjalnego powiadomienia

W celu stworzenia specjalnego powiadomienia, pracownik musi na stronie z powiadomieniami kliknąć przycisk „Stwórz powiadomienie”, które pokaże okienko, gdzie trzeba wpisać treść powiadomienia i do kogo jest ono adresowane (użytkownicy lub tylko pracownicy, gdzie druga opcja jest dostępna tylko dla administratorów).

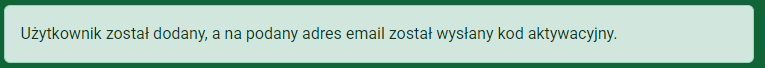
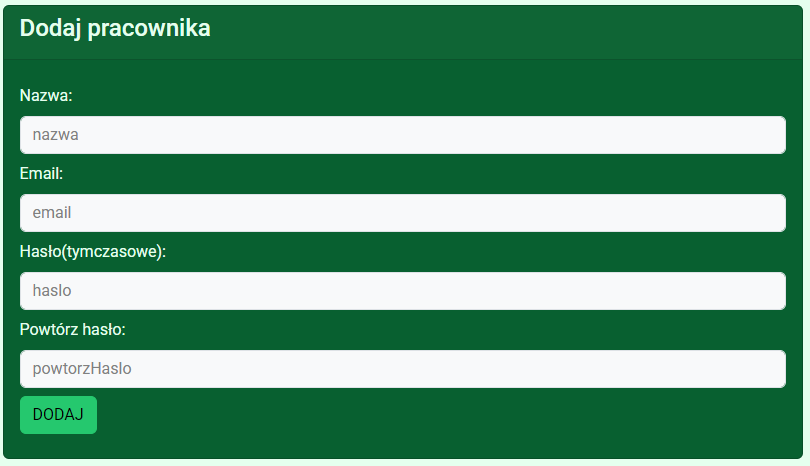


Rysunek Okno specjalnego powiadomienia i stworzone powiadomienie

### Funkcjonalności administratora

#### Tworzenie konta pracownika

W celu utworzenia pracownika, admin musi w panelu użytkowników kliknąć przycisk „Dodaj pracownika”, który przenosi go do strony, gdzie musi podać dane użytkownika, a po zatwierdzeniu wysyłana jest wiadomość na podany adres email (zob. rys. 90).



Rysunek Panel dodawania pracownika i komunikat po dodaniu

#### Banowanie pracowników

Banowanie pracowników działa dokładnie tak jak w rozdziale (Banowanie użytkowników), z tą różnicą, że tylko administrator ma do tego uprawnienia.

#### Wysyłanie powiadomień do pracowników

Administrator może wysyłać powiadomienia do pracowników, wybierając opcję odbiorcy jako pracowników w rozdziale Tworzenie specjalnego powiadomienia.

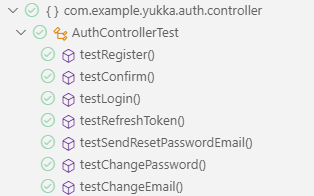
## Testy

W backendowej części aplikacji napisano testy integracyjne mające sprawdzać działanie programu w przypadkach, gdzie wymagany jest udział wielu komponentów. Testy integracyjne sprawdzają, jak różne części systemu współpracują ze sobą. Są one uruchamiane po pełnym załadowaniu kontekstu aplikacji i wykonywane są rzeczywiste interakcje pomiędzy komponentami.

Klasy testowe w tej aplikacji składają się z:

* użytkownika i jego autoryzacji,
* jednego lub więcej obiektów testowych,
* funkcji wykonywującej się przed wszystkimi testami, tworząca dane testowe w bazie,
* funkcji testowych wykonujących się w określonej kolejności,
* funkcji kończącej, mająca za zadanie wyczyszczenie bazy z danych testowych.

W panelu testowym można łatwo uruchamiać wybrane testy lub ich grupy. Każdy test posiada oznaczenie, czy się powiódł, a także ile czasu zajęło wykonanie danej funkcji. Przykładowo, *AuthControllerTest* obejmuje testy dotyczące *AuthController*, gdzie testuje się kolejno rejestrację, aktywację konta, logowanie, odświeżanie tokenu oraz zmianę hasła/emaila.



Rysunek Lista testów AuthControllerTest

Przykładowo, funkcja *testLogin()* (zob. rys. 92) testuje logowanie użytkownika, gdzie tworzone jest żądanie z danymi użytkownika zdefiniowanego w klasie testowej, który jest wysyłany do kontrolera, a potem sprawdza się poprawność odpowiedzi.



Rysunek Test logowania

## Podsumowanie

Aplikacja internetowa „Wirtualny Ogród Yukka” jest aplikacją internetową umożliwiającą zarządzanie wirtualnym ogrodem oraz interakcję społecznościową między użytkownikami.

**Spełnione funkcjonalności:**

* zarządzanie roślinami w oficjalnym katalogu bądź tworzenie własnych, pozwalając na dużą swobodę w dostosowaniu ich cech,
* projektowanie ogrodu z własnymi działkami, gdzie zasadza się rośliny z katalogu lub własne, przydzielając im obszar i dostosowując wygląd,
* tworzenie postów, komentowanie, ocenianie oraz prowadzenie rozmów prywatnych,
* system powiadomień i zgłoszeń informujący użytkowników i pracowników o zdarzeniach w aplikacji,
* rejestracja, aktywacja konta, logowanie, zarządzanie kontem i informacjami profilowymi, a także banowanie użytkowników przez pracowników i administratorów.

Aplikacja zawiera testy integracyjne sprawdzające poprawność działania poszczególnych komponentów oraz ich współpracę. Testy obejmują m.in. rejestrację, logowanie, zarządzanie roślinami, ogrodem, postami i komentarzami.

Planowany rozwój obejmuje dalsze usprawnienia interfejsu użytkownika (w szczególności ogrodu), zwiększyć możliwości modyfikacji roślin na działce, optymalizację wydajności, poprawę prowadzenia rozmów prywatnych poprzez implementację websocketów w celu prowadzenia rozmów w czasie rzeczywistym, dodanie galerii zdjęć użytkownika, rozszerzenie postów poprzez możliwość dodawania wielu obrazów naraz, a także optymalizacja pobierania komentarzy poprzez użycie paginacji albo Infinite Scrolling, mając również na uwadze ilość zagnieżdżonych odpowiedzi.

**Literatura**

|  |  |
| --- | --- |
| [1] | „Dokumentacja do Frameworka Angular 18,” [Online]. Available: https://angular.dev/overview. |
| [2] | S. Joshi, Full Stack Development with Angular and Spring Boot, BPB Publications, 2024. |
| [3] | J. S. Jensen, The Missing Bootstrap 5 Guide. Customize and extend Bootstrap 5 with Sass and JavaScript to create unique website designs, Packt Publishing, 2022. |
| [4] | „Dokumentacja Bootstrap 5,” [Online]. Available: https://getbootstrap.com/docs/5.0/getting-started/introduction/. |
| [5] | e. a. Kathy Sierra, Java. Rusz głową! Wydanie III, Helion, 2023. |
| [6] | e. a. Greg L. Turnquist, Learning Spring Boot 3.0. Simplify the development of production-grade applications using Java and Spring - Third Edition, Packt Publishing, 2022. |
| [7] | „Spring Boot Reference Documentation,” [Online]. Available: https://docs.spring.io/spring-boot/docs/3.2.5/reference/htmlsingle/. |
| [8] | „Dokumentacja Neo4j,” [Online]. Available: https://neo4j.com/docs/. |
| [9] | R. V. Bruggen, Learning Neo4j. Run blazingly fast queries on complex graph datasets with the power of the Neo4j graph database, Packt Publishing, 2014. |
| [10] | P. Stevens, UML. Inżynieria oprogramowania. Wydanie II, Helion, 2007. |
| [11] | A. S. e. al, Modelowanie systemów informatycznych w języku UML 2.1, Wydawnictwo Naukowe PWN, 2020. |
| [12] | e. a. Len Bass, Architektura oprogramowania w praktyce. Wydanie IV, Helion, 2022. |
| [13] | M. Masse, REST API Design Rulebook. Designing Consistent RESTful Web Service Interfaces, O'Reilly Media, 2011. |
| [14] | R. Mitchell, Web Scraping with Python. Collecting More Data from the Modern Web. 2nd Edition, O'Reilly Media, 2018. |
| [15] | „Dokumentacja dla Spring Data Neo4j,” [Online]. Available: https://spring.io/projects/spring-data-neo4j. |
| [16] | T. Marrs, JSON at Work. Practical Data Integration for the Web, O'Reilly Media, 2017. |
| [17] | „Dokumentacja Swagger,” [Online]. Available: https://swagger.io/docs/. |
| [18] | e. a. Aristeidis Bampakos, Poznaj Angular. Rzeczowy przewodnik po tworzeniu aplikacji webowych z użyciem frameworku Angular 15. Wydanie IV, Helion, 2023. |
| [19] | „ng-openapi-gen: An OpenAPI 3 code generator for Angular,” [Online]. Available: https://www.npmjs.com/package/ng-openapi-gen. |
| [20] | e. l. Lamis Chebbi, Reactive Patterns with RxJS and Angular Signals., Packt Publishing, 2024. |

**Spis ilustracji**

[Rysunek 1 Diagram przypadków użycia 7](#_Toc188608849)

[Rysunek 2 Widok architektury aplikacji 10](#_Toc188608850)

[Rysunek 3 Funkcja get\_all\_plant\_info z pliku scrap\_threads.py 13](#_Toc188608851)

[Rysunek 4 Węzeł Roslina i węzeł Roslina z etykietą RoslinaWlasna 14](#_Toc188608852)

[Rysunek 5 Węzeł Cecha i węzeł Cecha z etykietą CechaWlasna 16](#_Toc188608853)

[Rysunek 6 Węzeł ogrodu 16](#_Toc188608854)

[Rysunek 7 Węzeł Dzialka 17](#_Toc188608855)

[Rysunek 8 Węzeł Uzytkownik oraz węzły z etykietami Pracownik i Admin 18](#_Toc188608856)

[Rysunek 9 Węzeł Ustawienia 19](#_Toc188608857)

[Rysunek 10 Węzeł Post 19](#_Toc188608858)

[Rysunek 11 Węzeł Komentarz 20](#_Toc188608859)

[Rysunek 12 Węzeł RozmowaPrywatna 20](#_Toc188608860)

[Rysunek 13 Węzeł Powiadomienie bez i z etykietą Zgloszenie 21](#_Toc188608861)

[Rysunek 14 Relacja ZASADZONA\_NA 25](#_Toc188608862)

[Rysunek 15 Relacja OCENIL 25](#_Toc188608863)

[Rysunek 16 Relacja POWIADAMIA 26](#_Toc188608864)

[Rysunek 17 Widok bazy danych 26](#_Toc188608865)

[Rysunek 18 Struktura backendu 27](#_Toc188608866)

[Rysunek 19 Widok kodu źródłowego klasy Ogrod 29](#_Toc188608867)

[Rysunek 20 Żądania dla kontrolera Post w Swaggerze 29](#_Toc188608868)

[Rysunek 21 Widok żądania GET /posty 30](#_Toc188608869)

[Rysunek 22 Odpowiedź żądania 31](#_Toc188608870)

[Rysunek 23 Kod funkcji w kontrolerze 31](#_Toc188608871)

[Rysunek 24 Kod funkcji serwisu 32](#_Toc188608872)

[Rysunek 25 Zapytanie do bazy danych 32](#_Toc188608873)

[Rysunek 26 Odpowiedź żądania pobrania postów 33](#_Toc188608874)

[Rysunek 27 Folder z logami 33](#_Toc188608875)

[Rysunek 28 Przykład logów informacyjnego, ostrzeżenia i błędu 33](#_Toc188608876)

[Rysunek 29 Kod codziennego sprawdzania banów 34](#_Toc188608877)

[Rysunek 30 Struktura frontendu 35](#_Toc188608878)

[Rysunek 31 Komponent roslina-card oraz jego widok 36](#_Toc188608879)

[Rysunek 32 Kod źródłowy komponentu roslina-card.component.ts 37](#_Toc188608880)

[Rysunek 33 Kod źródłowy komponentu roslina-card.component.html 37](#_Toc188608881)

[Rysunek 34 Fragment kodu tworzący karty roślin 37](#_Toc188608882)

[Rysunek 35 Fragment kodu źródłowego pliku authentication.service.ts 38](#_Toc188608883)

[Rysunek 36 funkcja login z pliku login.ts 38](#_Toc188608884)

[Rysunek 37 Panel logowania i rejestracji 39](#_Toc188608885)

[Rysunek 38 Panel aktywacji i kod z wiadomości email 39](#_Toc188608886)

[Rysunek 39 Widok po zalogowaniu się 40](#_Toc188608887)

[Rysunek 40 Przejście do ustawień i do zmiany avatara 40](#_Toc188608888)

[Rysunek 41 Panel zmiany avatara i profil po zmianie 40](#_Toc188608889)

[Rysunek 42 Panel edycji profilu przed i po podaniu danych 41](#_Toc188608890)

[Rysunek 43 Profil po edycji 41](#_Toc188608891)

[Rysunek 44 Panel ustawień konta 42](#_Toc188608892)

[Rysunek 45 Panel zmiany adresu email 42](#_Toc188608893)

[Rysunek 46 Link do zmiany hasła i formularz 43](#_Toc188608894)

[Rysunek 47 Formularz zmiany hasła 43](#_Toc188608895)

[Rysunek 48 Panel usuwania konta 43](#_Toc188608896)

[Rysunek 49 Widok katalogu roślin 44](#_Toc188608897)

[Rysunek 50 Lista cech typu Gleba przed i po wyszukaniu 44](#_Toc188608898)

[Rysunek 51 Widok strony rośliny 45](#_Toc188608899)

[Rysunek 52 Wyszukiwarka postów , panel dodawania postu i przykładowy post 46](#_Toc188608900)

[Rysunek 53 Widok komentarzy postu 47](#_Toc188608901)

[Rysunek 54 Przejście do postów użytkownika 47](#_Toc188608902)

[Rysunek 55 Formularz dodania postu i post po dodaniu 48](#_Toc188608903)

[Rysunek 56 Usuwanie postu 48](#_Toc188608904)

[Rysunek 57 Dodawanie komentarza i dodany komentarz 48](#_Toc188608905)

[Rysunek 58 Edycja komentarza i komentarz po edycji 49](#_Toc188608906)

[Rysunek 59 Dodawanie odpowiedzi i odpowiedź po edycji 49](#_Toc188608907)

[Rysunek 60 Ocena postu/komentarza 49](#_Toc188608908)

[Rysunek 61 Ikona usuwania 49](#_Toc188608909)

[Rysunek 62 Lista użytkowników i profil jednego z nich 50](#_Toc188608910)

[Rysunek 63 Widok ogrodu użytkownika 50](#_Toc188608911)

[Rysunek 64 Widok działki 51](#_Toc188608912)

[Rysunek 65 Stworzony obraz działki 51](#_Toc188608913)

[Rysunek 66 Zmiana nazwy ogrodu 52](#_Toc188608914)

[Rysunek 67 Przycisk "Dodaj do działki" 52](#_Toc188608915)

[Rysunek 68 Strona dodawania rośliny do działki i nowo dodana roślina 52](#_Toc188608916)

[Rysunek 69 Przycisk przenoszenia do innej działki i widok 53](#_Toc188608917)

[Rysunek 70 Opcje zmiany właściwości rośliny 54](#_Toc188608918)

[Rysunek 71 Okno z wybieraniem koloru i roślina po zmianie 54](#_Toc188608919)

[Rysunek 72 Zmiana pozycji rośliny 55](#_Toc188608920)

[Rysunek 73 Usuwanie rośliny z działki 55](#_Toc188608921)

[Rysunek 74 Przejście do rozmów i rozmowy prywatne 56](#_Toc188608922)

[Rysunek 75 Przycisk "Zaproś" i wysłane zaproszenie 56](#_Toc188608923)

[Rysunek 76 Widok rozmowy prywatnej 57](#_Toc188608924)

[Rysunek 77 Dodanie wiadomości 57](#_Toc188608925)

[Rysunek 78 Potwierdzenie blokowania i wiadomość po zablokowaniu 58](#_Toc188608926)

[Rysunek 79 Okno powiadomień 58](#_Toc188608927)

[Rysunek 80 Panel powiadomień pracownika 59](#_Toc188608928)

[Rysunek 81 Opis zgłoszenia i widok wysłanego zgłoszenia 59](#_Toc188608929)

[Rysunek 82 Przycisk dodawania rośliny i fragment formularza 60](#_Toc188608930)

[Rysunek 83 Nowa cecha rośliny 61](#_Toc188608931)

[Rysunek 84 Dodana roślina 61](#_Toc188608932)

[Rysunek 85 Zmiana obrazu rośliny 62](#_Toc188608933)

[Rysunek 86 Potwierdzenie usuwania rośliny 62](#_Toc188608934)

[Rysunek 87 Formularz banowania i użytkownik po zbanowaniu 62](#_Toc188608935)

[Rysunek 88 Komunikat o zbanowaniu przy próbie zalogowania się 63](#_Toc188608936)

[Rysunek 89 Okno specjalnego powiadomienia i stworzone powiadomienie 63](#_Toc188608937)

[Rysunek 90 Panel dodawania pracownika i komunikat po dodaniu 64](#_Toc188608938)

[Rysunek 91 Lista testów AuthControllerTest 65](#_Toc188608939)

[Rysunek 92 Test logowania 65](#_Toc188608940)

**Streszczenie**

**Aplikacja internetowa do zarządzania ogrodem z elementami społecznościowymi**

**(Web Application for Garden Management with Social Elements)**

Celem pracy jest stworzenie aplikacji internetowej do zarządzania ogrodem z elementami społecznościowymi, zawierającą zarządzanie roślinami ze zróżnicowanymi cechami, projektowanie ogrodu podzielonego na działki w postaci siatek kafelków, na których można sadzić rośliny i je modyfikować, a także stworzenie części społecznościowej w formie postów, komentarzy, rozmów prywatnych, powiadomień o wydarzeniach, a także możliwości moderacji społeczności przy użyciu zgłoszeń i banowania. Wymienione cele zostały zrealizowane. Praca zawiera dokładny opis struktury aplikacji, procesu implementacji oraz przykłady użycia funkcjonalności.

1. Scrapper – narzędzie lub skrypt pobierający i przetwarzający dane ze stron internetowych poprzez analizę ich struktury HTML. [↑](#footnote-ref-1)
2. REST API – Architektoniczny styl projektowania interfejsów, który wykorzystuje protokół HTTP do komunikacji między klientem a serwerem zgodnie z wyznaczonymi przez niego zasadami. [↑](#footnote-ref-2)
3. CYPHER – język zapytań dla grafowych baz danych tj. Neo4j. [↑](#footnote-ref-3)
4. Tag – Tag Swaggera w przypisie służy do kategoryzowania endpointów API, grupując je w jeden rozwijalny dropdown w widoku Swaggera. [↑](#footnote-ref-4)
5. 200 – Kod HTML oznaczający pomyślne wykonanie operacji. [↑](#footnote-ref-5)
6. Route – Ścieżki do stron w formie adresów. Mogą zawierać parametry, np. id elementu. [↑](#footnote-ref-6)
7. base64 – kodowanie binarnych danych w formacie tekstowym, umożliwiający ich osadzanie w dokumentach HTML, CSS lub JSON bez bezpośredniego użycia plików. [↑](#footnote-ref-7)
8. HttpResponse – Odpowiedź http zawierająca kod statusu, nagłówki, i opcjonalne ciało. [↑](#footnote-ref-8)
9. Opcjonalna zawartość ciała odpowiedzi HTTP, w którym znajduje się wynik żądania. [↑](#footnote-ref-9)