

**UNIWERSYTET RZESZOWSKI**

**Kolegium Nauk Przyrodniczych**

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 4](#_Toc187618866)

[2 Cel i zakres pracy 5](#_Toc187618867)

[3 Funkcjonalności 5](#_Toc187618868)

[3.1 Narzędzia i technologie użyte w pracy 5](#_Toc187618869)

[3.1.1 Narzędzia użyte do części frontendowej 5](#_Toc187618870)

[3.1.2 Narzędzia użyte do części backendowej 6](#_Toc187618871)

[3.2 Diagram przypadków użycia (UML) 6](#_Toc187618872)

[4 Implementacja aplikacji 7](#_Toc187618873)

[4.1 Scrapper 7](#_Toc187618874)

[4.1.1 Pobranie roślin 7](#_Toc187618875)

[4.1.2 Pobranie obrazów roślin 7](#_Toc187618876)

[4.2 Struktura bazy danych 9](#_Toc187618877)

[4.2.1 Źródła danych 9](#_Toc187618878)

[4.2.2 Węzły, etykiety i relacje bazy danych 10](#_Toc187618879)

[4.3 Struktura backendu aplikacji 21](#_Toc187618880)

[4.3.1 Przykładowa klasa modelu danych 23](#_Toc187618881)

[4.3.2 Przykład procesu żądania i odpowiedzi 24](#_Toc187618882)

[4.3.3 Logowanie zdarzeń i błędów 28](#_Toc187618883)

[4.3.4 Zadania okresowe w tle 29](#_Toc187618884)

[4.4 Struktura frontendu aplikacji 29](#_Toc187618885)

[4.4.1 Przykładowy komponent 30](#_Toc187618886)

[4.4.2 Przykład metody API z serwisu 32](#_Toc187618887)

[4.4.3 Seedery 32](#_Toc187618888)

[5 Opis funkcjonalności systemu 34](#_Toc187618889)

[5.1 Funkcjonalności zwykłego użytkownika 34](#_Toc187618890)

[5.1.1 Rejestracja 34](#_Toc187618891)

[5.1.2 Przeglądanie roślin w katalogu 35](#_Toc187618892)

[5.1.3 Przeglądanie postów i komentarzy 37](#_Toc187618893)

[5.1.4 Przeglądanie użytkowników 38](#_Toc187618894)

[5.1.5 Przeglądanie ogrodów 38](#_Toc187618895)

[5.2 Funkcjonalności pracownika(TODO) 40](#_Toc187618896)

[5.2.1 Banowanie użytkowników 40](#_Toc187618897)

[5.3 Funkcjonalności admina(TODO) 41](#_Toc187618898)

[5.4 Zarządzanie zasobami 41](#_Toc187618899)

[5.4.1 Zarządzanie kontem 41](#_Toc187618900)

[5.4.2 Zarządzanie roślinami 46](#_Toc187618901)

[5.4.3 Zarządzanie własnymi roślinami 48](#_Toc187618902)

[5.4.4 Zarządzanie postami i komentarzami 48](#_Toc187618903)

[5.4.5 Zarządzanie powiadomieniami 52](#_Toc187618904)

[5.4.6 Zarządzanie rozmowami 55](#_Toc187618905)

[5.4.7 Zarządzanie ogrodem 58](#_Toc187618906)

[6 Testy(TODO) 62](#_Toc187618907)

1. [Podsumowanie(TODO) 64](#_Toc187618908)
2. [Streszczenie(TODO) 64](#_Toc187618909)
3. [Spis ilustracji 65](#_Toc187618910)

## Wstęp

Aplikacja internetowa pt. „Wirtualny Ogród Yukka” jest m systemem umożliwiający 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) w ogrodzie, 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. Jednym z tych sposobów jest 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. W razie, gdyby dwójka użytkowników chciała porozmawiać na cztery oczy, możliwe jest prowadzenie rozmowy prywatnej. W razie, gdyby jakiś użytkownik zachowywał się niepoprawnie, możliwe jest zgłoszenie go przez innych użytkowników i późniejsze zbanowanie go przez pracowników. Dzięki temu system wspiera zarówno początkujących, jak i zaawansowanych ogrodników, tworząc przyjazne środowisko do dzielenia się swoją pasją.

## Cel i zakres pracy

## Funkcjonalności

### Narzędzia i technologie użyte w pracy

#### Narzędzia użyte do części frontendowej

**TypeScript**

TypeScript to rozszerzenie języka JavaScript dodające typowanie statyczne, klasy oraz interfejsy. Ma dużo mniejszą tolerancję na błędy, a sam jego kod jest kompilowany na końcu do JavaScript. Używany w wersji 5.5.2 w Angularze do programowania części frontendowej. Można ją pobrać z tego linku: <https://www.typescriptlang.org/download/>

**Angular**

Jest to framework opartym na języku TypeScript do budowy aplikacji webowych, rozwijanego przez Google. Wspomniana wersja posiada mechanikę sygnałów do komunikacji pomiędzy komponentami, wsparcie dla SSR (rendererowania po stronie serwera), hydracji (użycie statycznej części wygenerowanej strony przez SSR), oraz lepsza obsługa ładowania asynchronicznego. Używany jest w wersji 18.2.11. Można go pobrać z linku: <https://v18.angular.dev/installation>

**Pozostałe narzędzia i technologie:**

**Bootstrap 5** - biblioteka CSS. Zawiera zestaw narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Bazuje głównie na gotowych rozwiązaniach HTML oraz CSS i może być stosowany m.in. do stylizacji poszczególnych elementów strony.

**Npm (Node Package Manager**) - Menedżer pakietów dla JavaScriptu, dostarczany wraz z Node.js. Służy do instalowania, zarządzania i udostępniania bibliotek oraz narzędzi JavaScript.

#### Narzędzia użyte do części backendowej

**Java**

Java to język obiektowy działający na różnych systemach operacyjnych bez większych trudów. Jest szeroko stosowana do tworzenia przeróżnych aplikacji czy to internetowych, czy to mobilnych. Używana jest wersja 17. Można ją pobrać z linku: <https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>

**Spring Boot**

Spring Boot to 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. Używany w wersji 3.3.2. Można go pobrać z linku: <https://docs.spring.io/spring-boot/installing.html>

### Diagram przypadków użycia (UML)

1. (Konsultacja z Kamil Matusz)

Uprawnienia użytkownika:

* Tworzenie własnych roślin i ich modyfikacja
* Dodawanie roślin do swojego ogrodu i ich modyfikacja
* Tworzenie postów i komentarzy
* Zapraszanie i uczestniczenie w rozmowach prywatnych
* Zgłaszanie i blokowanie użytkowników
* Zmiana nazwy ogrodu, działki, informacji profilowych
* Zmiana własnych ustawień, np. wyłączenie powiadomień o odpowiedziach do własnych komentarzy, czy wyłączenie powiadomień okresie owocowania roślin na działce
* Usuwanie własnych roślin, komentarzy i postów

Uprawnienia pracownika:

* Tworzenie roślin do katalogu i ich modyfikacja
* Dostęp do treści prywatnych zwykłych użytkowników
* Usuwanie treści zwykłych użytkowników
* Banowanie zwykłych użytkowników

Uprawnienia admina:

* Dostęp do treści prywatnych użytkowników i pracowników
* Usuwanie treści użytkowników i pracowników
* Banowanie pracowników

## Implementacja aplikacji

### Scrapper

Jest to scrapper napisany w języku Python używany do pobierania danych o roślinach ze strony <https://e-katalogroslin.pl> do pliku *katalog\_roslin.csv.* Wykorzystuje on sterownik Firefox, wielowątkowość oraz asynchroniczność do szybszego pobierania danych ze stron.

#### Pobranie roślin

Aby rozpocząć proces pobierania roślin, należy uruchomić plik scrap\_threads.py, który pobierze dane od początkowej strony aż do ostatniej. Należy mieć na uwadze, że czasami program może mieć trudności z dostępnością do strony, przejściem na inną stronę oraz inne czynniki zewnętrzne które mogą zakłócić bądź uniemożliwić scrappowanie danych. Pobranie wszystkich danych ze strony powinno zająć ok. 50-70 minut.

#### Pobranie obrazów roślin

Plik *katalog\_roslin.csv* zawiera informacje o roślinach takie jak nazwa rośliny, nazwa łacińska, opis, kwasowość gleby, a także link do obrazu rośliny (jeśli takowy posiada). Aby pobrać obrazy rośliny, należy uruchomić plik *download\_images.py*, który będzie iterował przez kolejne rzędy pliku *katalog\_roslin.csv* i zapisze obrazy do folderu *images.* Proces ten powinien zająć do 5 minut.

Oto fragment kodu, w którym wykonywana jest pętla scrappowania kolejnych stron. Funkcja get\_all\_plant\_info pobiera informacje o roślinach z podanej listy adresów URL. Używa biblioteki Selenium do nawigacji po stronach internetowych i oczekiwania na załadowanie elementów. Po przetworzeniu strony próbuje przejść do następnej sprawdzając, czy adres URL zmienił się i próbuje ponownie, jeśli nie. Zbiera informacje o roślinach i zwraca je jako listę. Na koniec zamyka przeglądarkę. Ta funkcja jest przydzielana dla każdego wątku, aby przyśpieszyć zbieranie danych.

async def get\_all\_plant\_info(start\_url, url\_list, amount=9999):

    max\_tries = 5

    page\_number = 1

    all\_plant\_info = []

    current\_url = start\_url

    driver = webdriver.Firefox(options=firefox\_options)

    driver.implicitly\_wait(10)

    driver.get(start\_url)

    thread\_id = threading.get\_ident()

    try:

        WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.ID, "listning")))

    except:

        print(f"Timeout! Nie udało się załadować sekcji listingu dla {start\_url}.")

        driver.quit()

        return None

    async with aiohttp.ClientSession() as session:

        while current\_url != last\_url and amount != 0:

            if current\_url in url\_list and current\_url != start\_url:

                print(f"Zatrzymanie wątku, napotkano adres startowy innego wątku: {current\_url}")

                break

            plant\_info\_list = await parse\_page(driver.page\_source)

            tasks = [asyncio.create\_task(get\_plant\_properties(session, plant['link'])) for plant in plant\_info\_list]

            properties\_list = await asyncio.gather(\*tasks)

            updated\_plant\_info\_list = []

            for plant, properties in zip(plant\_info\_list, properties\_list):

                if  properties.get('opis') is None:

                    continue

                plant.update(properties)

                del plant['link']

                updated\_plant\_info\_list.append(plant)

            plant\_info\_list = [plant for plant in updated\_plant\_info\_list if plant.get('docelowa\_wysokosc')]

            all\_plant\_info.extend(plant\_info\_list)

            if(page\_number % 10 == 0):

                print(f"Wątek [{thread\_id}]: Przetworzone strony: [{page\_number}]. Pobrano dane z: {current\_url}")

            for \_ in range(max\_tries):

                try:

                    next\_button = WebDriverWait(driver, 12).until(EC.element\_to\_be\_clickable((By.XPATH, "//button[@data-next]")))

                    next\_button.click()

                    WebDriverWait(driver, 12).until(EC.url\_changes(current\_url))

                    break

                except (ElementClickInterceptedException, StaleElementReferenceException, TimeoutException):

                    continue

            try:

                WebDriverWait(driver, 7).until(EC.presence\_of\_element\_located((By.ID, "listning")))

            except:

                print("Nie udało się załadować nowej strony.")

            new\_url = driver.current\_url

            tries = 0

            if new\_url == current\_url:

                print("URL nie zmienił się. Spróbuj ponownie.")

                tries += 1

                if tries >= 30:

                    print("Przekroczono limit prób zmiany adresu.")

                    break

                continue

            if tries > 0:

                print(f"Ilość prób zmiany adresu nr: {tries}.")

            current\_url = new\_url

            page\_number += 1

            amount -= 1

    driver.quit()

    return all\_plant\_info

Równanie 1 Funkcja get\_all\_plant\_info z pliku scrap\_threads.py

### Struktura bazy danych

**Neo4j**

Neo4j jest grafową bazą 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. Neo4j używany jest w wersji 5.12.0.

#### Źródła danych

Źródłem danych jest strona internetowa *e-katalogroslin.pl* należąca do Związku Szkółkarzy Polskich, który jest stowarzyszeniem producentów drzew, krzewów i bylin, a także roślin owocowych. Wymieniona strona jest katalogiem zawierający ponad 10 000 roślin o różnych właściwościach takich jak preferowany rodzaj gleby, okres owocowania, zimozieloność liści, zakres wysokości itp.

Link do strony: <https://e-katalogroslin.pl>

#### Węzły, etykiety i relacje bazy danych

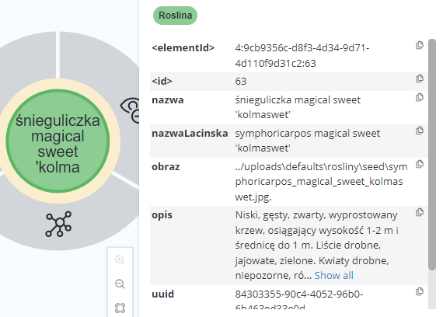
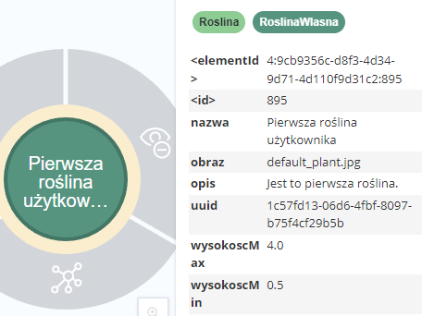
##### Węzły

###### Roslina, RoslinaWlasna

Roślina jest jednym z głównym węzłów w bazie danych. Reprezentuje roślinę z katalogu oraz roślinę użytkownika i jest wykorzystywana w ogrodzie. Rośliny użytkownika nienależące do oficjalnego katalogu posiadają etykietę Roslina oraz RoslinaWlasna, a roślina z katalogu może mieć tylko etykietę Roslina.

**Właściwości:**

* 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

Rysunek 0.1 Węzeł Roslina i węzeł Roslina z etykietą RoslinaWlasna

###### Cecha

Rośliny posiadają 16 cech opisanych etykietami takimi jak *Owoc, Grupa, Gleba,* które klasyfikują typ danej właściwości. Zwykła cecha ma etykietę *Cecha*, za to właściwość użytkownika ma dodatkowo etykietę *CechaWlasna.*

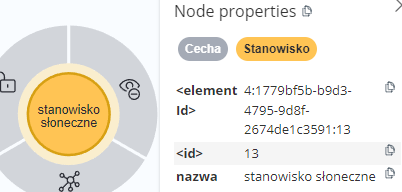
**Etykiety główne:** Cecha, CechaWlasna

**Etykiety dodatkowe:**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Opis** | **Zdjęcie** |
| Forma | Forma rośliny |  |
| Gleba | Rodzaj gleby przyjazny dla rośliny |  |
| Grupa | Grupa rośliny |  |
| Kolor | Kolor kwiatów albo liści rośliny |  |
| Kwiat | Rodzaj kwiatu rośliny |  |
| Odczyn | Odczyn ph gleby przyjazny dla rośliny |  |
| Okres | Okres owocowania albo kwitnienia rośliny wyrażony w miesiącach |  |
| Owoc | Rodzaj owocu rośliny |  |
| Podgrupa | Podgrupa rośliny |  |
| Pokroj | Pokrój lub kształt rośliny |  |
| SilaWzrostu | Siła wzrostu rośliny, czyli jak szybko rośnie |  |
| Stanowisko | Odpowiednie nasłonecznienie rośliny |  |
| Wilgotnosc | Optymalna wilgotność gleby rośliny |  |
| Zastosowanie | Do czego pasuje roślina, albo czy ma jakiś użytek poza wyglądem |  |
| Zimozielonosc | Jak zachowują się liście/igły rośliny podczas zimy, czy opadają, a może nie) |  |

**Właściwości:**

* nazwa – nazwa cechy



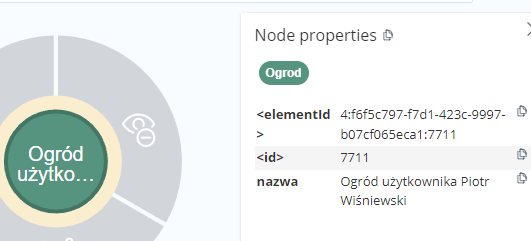
Rysunek 0.2 Węzeł Cecha i węzeł Cecha z etykietą CechaWlasna

###### Ogrod

Ogród jest węzłem tworzonym dla każdego użytkownika. Reprezentuje ogród podzielony na 10 działek, które również są węzłami złączonymi z nim.

**Właściwości:**

* nazwa – nazwa ogrodu



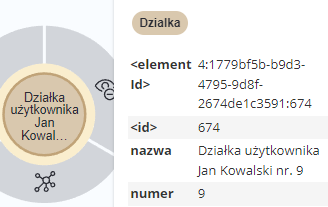
Rysunek 0.3 Węzeł ogrodu

###### Dzialka

Działka jest węzłem związanym z ogrodem, tworzony wraz z nim. Łącznie dla każdego użytkownika jest tworzone po 10 działek na ogród, gdzie każda działka ma wymiary 20x20 kafelków.

**Właściwości:**

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



Rysunek 0.4 Węzeł Dzialka

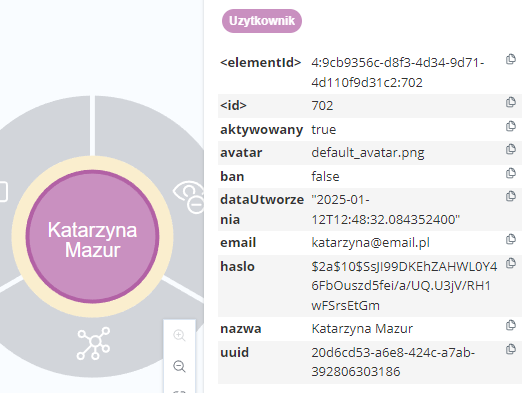
###### Uzytkownik, Pracownik, Admin

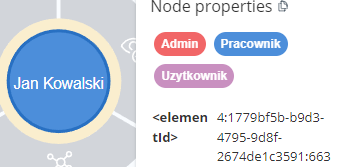
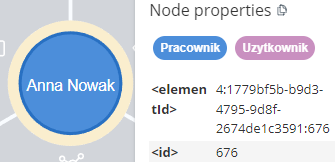
Użytkownik jest węzłem pełniącym rolę użytkownika w bazie danych. Zwykli użytkownicy posiadają tylko etykietę Uzytkownik, za to pracownicy posiadają dodatkowo etykietę Pracownik, a admini posiadają wszystkie z tych etykiet. Każda etykieta oznacza inny poziom uprawnień.

Dodatkowo, każdy użytkownik ma węzeł Ustawienia, węzeł Ogrod, oraz 10 węzłów Dzialka przypisanych do tego ogrodu. Informacje o użytkowniku są w ten sposób rozdzielone, aby zoptymalizować pobór danych z bazy danych i pobierać tylko te części danych, które są potrzebne.

**Właściwości:**

* 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





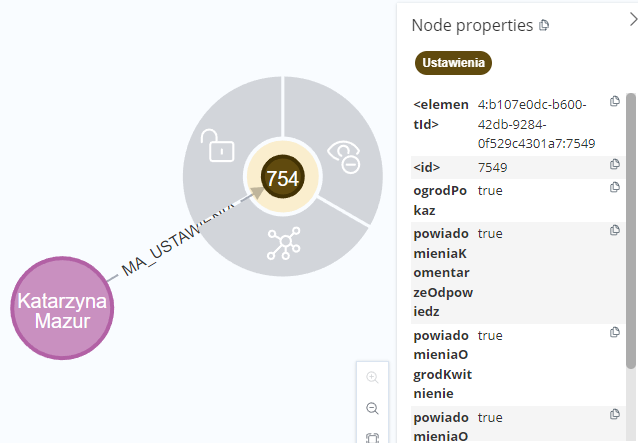
Rysunek 0.5 Węzeł Uzytkownik oraz węzły z etykietami Pracownik i Admin

###### Ustawienia

Każdy użytkownik ma swoje ustawienia, w których decyduje o powiadomieniach i widoczności swojego profilu.

**Właściwości:**

* ogrodPokaz – pozwala innym przeglądać ogród użytkownika (nie dotyczy pracowników i adminów)
* powiadomieniaKomentarzeOdpowiedz – powiadomienia o odpowiedziach na komentarze
* powiadomieniaOgrodKwitnienie – powiadomienia o kwitnieniu roślin w ogrodzie
* powiadomieniaOgrodOwocowanie – powiadomienia o owocowaniu roślin
* statystykiProfilu – widoczność statystyk na profilu



Rysunek 0.6 Węzeł Ustawienia

###### Post

Węzeł Post pełni rolę postu w części społecznościowej. Oprócz nazwy i opisu może również zawierać jeden obraz. Do tego postu mogą być przypisywane komentarze użytkowników, a także może być on oceniany przez nich.

**Właściwości:**

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



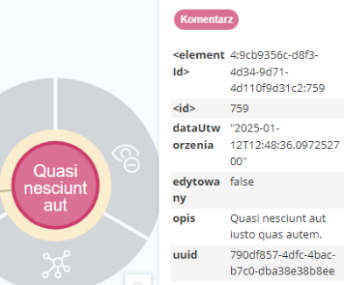
Rysunek 0.7 Węzeł Post

###### Komentarz

Węzeł Komentarz pełni rolę komentarza w części społecznościowej. Są one używane do komentowania postów, odpowiadania na komentarze, a także służą jako wiadomości w rozmowach prywatnych.

**Właściwości:**

* uuid – Identyfikator komentarza
* opis – treść komentarza
* obraz – ścieżka do obrazu komentarza
* edytowany – oświadcza, czy komentarz był edytowany
* dataUtworzenia – data utworzenia komentarza



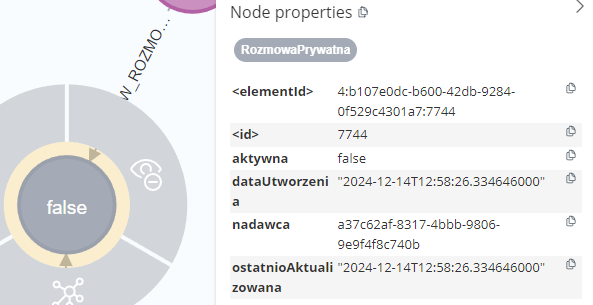
Rysunek 0.8 Węzeł Komentarz

###### RozmowaPrywatna

Węzeł RozmowaPrywatna pełni rolę rozmowy prywatnej pomiędzy dwoma użytkownikami w części społecznościowej. Aby rozmowa prywatna miała miejsce, użytkownik A musi zaprosić użytkownika B do rozmowy, a odbiorca musi ją zaakceptować. Wtedy możliwe jest pisanie wiadomości pomiędzy nimi.

**Właściwości:**

* aktywna – oznacza, czy rozmowa prywatna jest aktywna (odbiorca zaakceptował zaproszenie)
* nadawca – identyfikator nadawcy
* dataUtworzenia – data utworzenia rozmowy (wysłania zaproszenia)
* ostatnioAktualizowana – data od ostatniej wysłanej wiadomości, lub od kiedy rozmowa została zaakceptowana



Rysunek 0.9 Węzeł RozmowaPrywatna

###### Powiadomienie, Zgloszenie

Węzeł Powiadomienie powiadania użytkownika o różnych zdarzeniach w systemie, takie jak odpowiedź na komentarz, rozpoczęcie okresu owocowania roślin na działce, czy zaproszeniu przez użytkownika do rozmowy prywatnej. Jeśli posiada ono dodatkowo etykietę Zgloszenie, to pełni ono rolę zgłoszenia przez użytkownika, które jest przypisywane do pracownika o najmniejszej aktualnej liczbie zgłoszeń. Użytkownik może ustawić powiadomienie jako przeczytane, albo je usunąć pod warunkiem, że nie jest to zgłoszenie.

**Właściwości:**

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

Rysunek 0.10 Węzeł Powiadomienie bez i z etykietą Zgloszenie

##### Relacje

###### Relacje bez właściwości

|  |  |  |  |
| --- | --- | --- | --- |
| **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 prywatnej. |  |
| 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. |  |

###### Relacje z właściwościami

Relacja: ZASADZONA\_NA

Połączenie kierunku Roslina/RoslinaWlasna -> Dzialka. Wskazuje działkę, na której została zasadzona roślina.

**Właściwości:**

* x, y – pozycja rośliny.
* tabX, tabY – jednowymiarowe tablice kafelków (x, y); używane zamiast tablic wielowymiarowych (ograniczenie Neo4j).
* kolor – kolor kafelków w HEX.
* obraz – ścieżka do obrazu rośliny
* tekstura – ścieżka do tekstury kafelków
* wyswietlanie – tryb wyświetlania rośliny
  + - KOLOR –pokolorowane kafelki.
    - TEKSTURA – tekstura kafelków (wymaga tekstury rośliny).
    - TEKSTURA\_KOLOR – pokolorowane kafelki z teksturą.



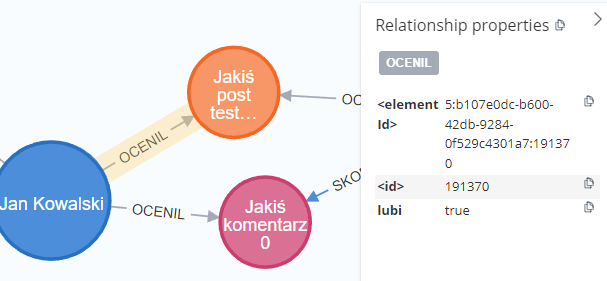
Rysunek 0.11 Relacja ZASADZONA\_NA

Relacja: OCENIL

Połączenie kierunku Uzytkownik -> Post albo Uzytkownik -> Komentarz. Wskazuje węzeł oceniony przez użytkownika, którym jest post albo komentarz, gdyż mogą one być oceniane przez niego.

**Właściwości:**

* lubi – ocena pozytywna albo negatywna



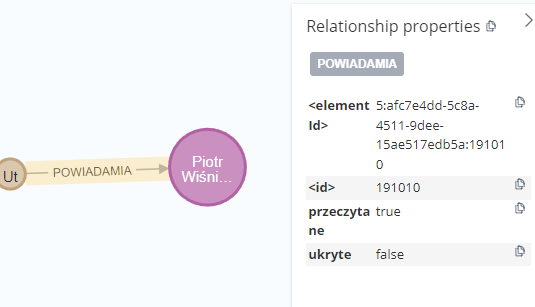
Rysunek 0.12 Relacja OCENIL

Relacja: POWIADAMIA

Połączenie kierunku Powiadomienie -> Uzytkownik. Wskazuje użytkownika, który jest powiadamiany.

**Właściwości:**

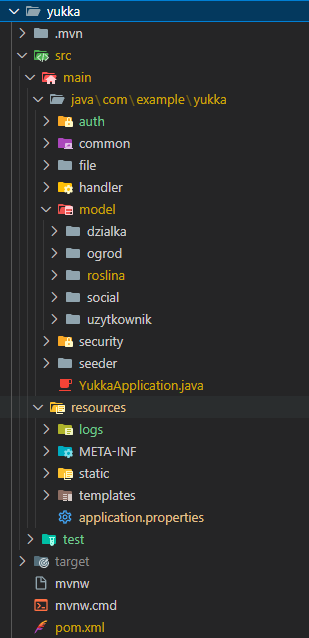
* przeczytane – status przeczytania powiadomienia.
* ukryte – status ukrycia powiadomienia przez użytkownika (dotyczy zgłoszeń i specjalnych powiadomień).



Rysunek 0.13 Relacja POWIADAMIA

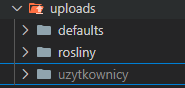
### Struktura backendu aplikacji

Backend, oparty na frameworku Spring Boot, obsługuje żądania REST od frontendu, waliduje dane, wykonuje zapytania do bazy danych i zwraca odpowiedzi w formacie JSON. Projekt podzielono na część programową (zawiera logikę aplikacji) i część testową(obejmuje testy jednostkowe weryfikujące działanie poszczególnych elementów).



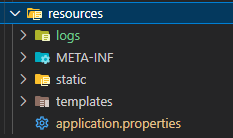
Rysunek 0.15 Struktura backendu

* Auth – Obsługa autoryzacji: logowanie, rejestracja, potwierdzenia e-mail, usuwanie konta.
* Common – Obiekty wspólne, np. odpowiedzi z paginacją.
* File – Zarządzanie plikami i folderami w katalogu uploads (pliki domyślne, rośliny, foldery użytkowników).



Rysunek 0.16 Katalog uploads

* 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).



Rysunek 0.17 Katalog resources

* Pom.xml – Plik z zależnościami programu

#### Przykładowa klasa modelu danych

##### Obiekt: Oceniany

Nie jest węzłem w bazie danych, lecz rodzicem dla obiektów (Post, Komentarz), które dziedziczą relację OCENIL (@Relationship). Dzięki temu posiadają metody do liczenia ocen.

@Getter

@Setter

@ToString

@NoArgsConstructor

@AllArgsConstructor

public class Oceniany {

    @Relationship(type = "OCENIL", direction = Relationship.Direction.INCOMING)

    private List<OcenilReverse> ocenil;

    public int getOcenyLubi() {

        return (int) ocenil.stream().filter(OcenilReverse::isLubi).count();

    }

    public int getOcenyNieLubi() {

        return (int) ocenil.stream().filter(ocenil -> !ocenil.isLubi()).count();

    }

}

Równanie 2 Kod obiektu Oceniany

##### Obiekt: OcenilReverse

Reprezentuje relację OCENIL z właściwościami:

* Lubi – pozytywna/negatywna ocena
* Uzytkownik – wskazywany węzeł Uzytkownik.

@RelationshipProperties

@Getter

@Builder

@AllArgsConstructor

@NoArgsConstructor

public class OcenilReverse {

    @Id @GeneratedValue

    private Long id;

    @Property("lubi")

    private boolean lubi;

    @TargetNode

    private Uzytkownik uzytkownik;

}

Równanie 3 Kod obiektu Ocenil

##### Obiekt: Post

Reprezentuje węzeł Post w bazie i dziedziczy właściwości Oceniany.

**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.
* @CreatedDate – Data utworzenia.

@Node

@Getter

@Setter

@ToString

@NoArgsConstructor

@AllArgsConstructor

@Builder

public class Post extends Oceniany {

    @Id @GeneratedValue

    private Long id;

    @Property(name = "uuid")

    private String uuid;

    @Property(name = "tytul")

    private String tytul;

    @Property(name = "opis")

    private String opis;

    @Property(name = "obraz")

    private String obraz;

    @CreatedDate

    @Property(name = "dataUtworzenia")

    private LocalDateTime dataUtworzenia;

    @Relationship(type = "MA\_POST", direction = Relationship.Direction.INCOMING)

    private Uzytkownik autor;

    @Relationship(type = "MA\_KOMENTARZ", direction = Relationship.Direction.OUTGOING)

    private List<Komentarz> komentarze;

    @Relationship(type = "JEST\_W\_POSCIE", direction = Relationship.Direction.INCOMING)

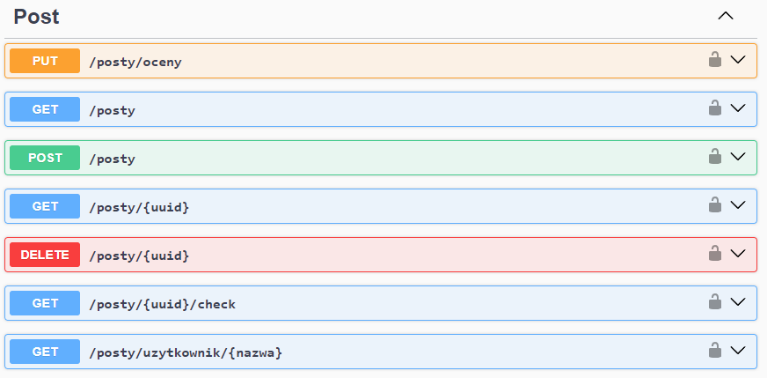
      private List<Komentarz> komentarzeWPoscie;

}

Figure 1 Kod obiektu Post

#### Przykład procesu żądania i odpowiedzi

Aby wykonał jakąś operacje, musi on wysłać żądanie do serwera na backendzie. Poniżej znajduje się widok możliwych żądań dla Kontrolera o swaggerowym tagu „Post”:

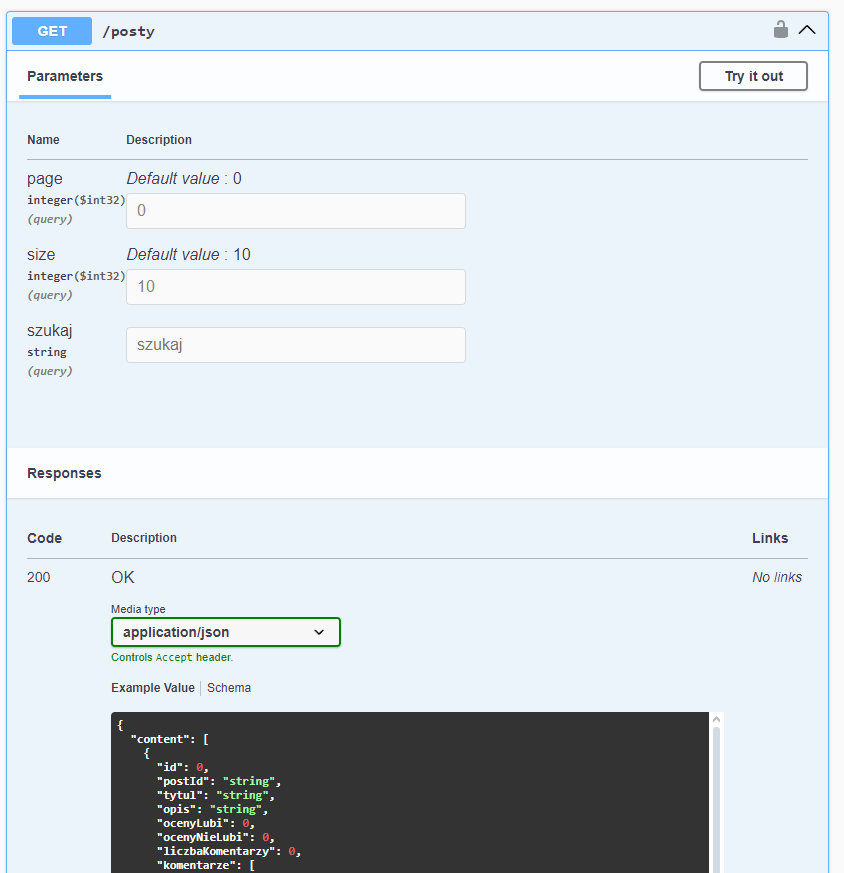


Rysunek 0.18 Żądania dla kontrolera Post w Swaggerze

Kontroler posiada następujące ścieżki:

* /posty/oceny PUT – Dodanie lub aktualizacja oceny postu.
* /posty GET – Lista postów z paginacją.
* /posty POST – Dodanie nowego postu.
* /posty/{uuid} GET – Szczegóły postu o podanym identyfikatorze.
* /posty/{uuid} DELETE – Usunięcie postu o podanym identyfikatorze.
* /posty/{uuid}/check GET – Post o podanym identyfikatorze, bez komentarzy.
* /posty/uzytkownik/{nazwa} – Lista postów użytkownika z paginacją.

Załóżmy, że użytkownik chce pobrać sobie wszystkie posty. Do tego zadania należy użyć metody GET /posty. Poniżej znajduje się widok ze Swaggera:



Rysunek 0.19 Widok żądania GET /posty

Do żądania można opcjonalnie podać następujące parametry:

* page(strona) o domyślnej wartości 0
* size(ilość zwracanych elementów na stronę) o domyślnej wartości 10
* szukaj(nazwa postu).

Przykładowa odpowiedź dla tego żądania zawiera listę postów wraz z takimi informacjami jak aktualna strona, całkowitą ilość stron, całkowitą ilość elementów, oraz czy jest to pierwsza/ostatnia strona:

"content": [

    {

      "id": 0,

      "uuid": "string",

      "tytul": "string",

      "opis": "string",

      "ocenyLubi": 0,

      "ocenyNieLubi": 0,

      "liczbaKomentarzy": 0,

      "komentarze": [

        {

          "id": 0,

          "uuid": "string",

          "opis": "string",

          "edytowany": true,

          "ocenyLubi": 0,

          "ocenyNieLubi": 0,

          "dataUtworzenia": "string",

          "post": "string",

          "odpowiadaKomentarzowi": "string",

          "odpowiedzi": [

            "string"

          ],

          "uzytkownikNazwa": "string",

          "obraz": "string",

          "avatar": "string"

        }

      ],

      "uzytkownik": "string",

      "obraz": "string",

      "avatar": "string",

      "dataUtworzenia": "string"

    }

  ],

  "number": 0,

  "size": 0,

  "totalElements": 0,

  "totalPages": 0,

  "first": true,

  "last": true

}

Równanie 4 Odpowiedź żądania

Samo żądanie w kodzie wygląda tak:

    @GetMapping(produces="application/json")

    public ResponseEntity<PageResponse<PostResponse>> findAllPosty(

            @RequestParam(name = "page", defaultValue = "0", required = false) int page,

            @RequestParam(name = "size", defaultValue = "10", required = false) int size,

            @RequestParam(name = "szukaj", required = false) String szukaj) {

        return ResponseEntity.ok(postService.findAllPosts(page, size, szukaj));

    }

Równanie 5 Kod funkcji w kontrolerze

Kontroler wykonuje funkcję z serwisu, którą opakowuje w ciało odpowiedzi o kodzie 200 (OK). Przechodząc do serwisu można znaleźć następujący kod:

    @Transactional(readOnly = true)

    public PageResponse<PostResponse> findAllPosts(int page, int size, String szukaj) {

        Pageable pageable = PageRequest.of(page, size, Sort.by("post.dataUtworzenia").descending());

        Page<Post> posts = postRepository.findAllPosts(szukaj, pageable);

        return postMapper.postResponsetoPageResponse(posts);

    }

Równanie 6 Kod funkcji serwisu

Serwis używa metody z repozytorium, aby zwróciło odpowiedź zawierającą daną stronę z paginacji postów, a następnie mapuje je odpowiednio do odpowiedzi, którą zwraca. Przechodząc do repozytorium, można znaleźć następujące zapytanie:

    @Query(value = """

        MATCH path = (post:Post)<-[:MA\_POST]-(:Uzytkownik)

        OPTIONAL MATCH path2 = (post)-[:OCENIL]-(:Uzytkownik)

        OPTIONAL MATCH path3 = (post)<-[:JEST\_W\_POSCIE]-(kom:Komentarz)

        WHERE $szukaj IS NULL OR post.tytul CONTAINS $szukaj OR post.opis CONTAINS $szukaj

        RETURN post, collect(nodes(path)), collect(relationships(path)),

                collect(nodes(path2)), collect(relationships(path2)),

                collect(nodes(path3)), collect(relationships(path3))

        :#{orderBy(#pageable)} SKIP $skip LIMIT $limit

        """,

       countQuery = """

        MATCH (post:Post)<-[:MA\_POST]-(:Uzytkownik)

        WHERE $szukaj IS NULL OR post.tytul CONTAINS $szukaj OR post.opis CONTAINS $szukaj

        RETURN count(post)

        """)

    Page<Post> findAllPosts(@Param("szukaj") String szukaj, Pageable pageable);

Równanie 7 Zapytanie do bazy danych

Zapytanie w języku CYPHER 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 (wymagane dla obiektu Page).

Po wykonaniu tych operacji, otrzymuje się odpowiedź, która może wyglądać przykładowo tak (z pominięciem zawartości obrazów, które są dużymi tablicami bajtów):

{

    "content": [

      {

        "id": 7615,

        "uuid": "760b84e6-2d16-4abf-9d11-e0951d6e4c0f",

        "tytul": "Ut molestiae id quisquam.",

        "opis": "Est deserunt dolor aspernatur. Molestias in nobis iste maiores dicta. Debitis porro laudantium dolores quibusdam autem reprehenderit pariatur.",

        "ocenyLubi": 4,

        "ocenyNieLubi": 1,

        "liczbaKomentarzy": 4,

        "komentarze": [],

        "uzytkownik": "Anna Nowak",

        "avatar": "…"

        "dataUtworzenia": "3 dni temu"

      },

      {

        "id": 7603,

        "uuid": "2870b5c0-0962-4a37-ad4b-fd9634c35765",

        "tytul": "Optio voluptate ut.",

        "opis": "Est eaque eligendi quo in iusto nihil. Impedit dolorum assumenda voluptatem reprehenderit repellendus architecto.",

        "ocenyLubi": 0,

        "ocenyNieLubi": 2,

        "liczbaKomentarzy": 6,

        "komentarze": [],

        "uzytkownik": "Jan Kowalski",

        "obraz": "…"

        "avatar": "…"

        "dataUtworzenia": "3 dni temu"

      }

    ],

    "number": 0,

    "size": 2,

    "totalElements": 5,

    "totalPages": 3,

    "first": true,

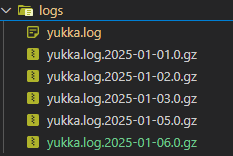
    "last": false

}

Równanie 8 Odpowiedź żądania

#### 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, gdzie przechowywane są logi do tygodnia wstecz:



Rysunek 0.20 Folder z logami

Przykładowy log informacyjny:

2025-01-12T13:59:14.233+01:00  INFO 1880 --- [yukka] [main] c.e.y.m.d.controller.DzialkaService      : Zmiana pozycji rośliny z [12, 11] na [3, 4] na działce [2] użytkownika Piotr Wiśniewski

Równanie 9 Log informacyjny

Przykładowy log ostrzeżenia (przestarzałą funkcjonalność):

2025-01-12T13:59:11.027+01:00  WARN 1880 --- [yukka] [main] o.s.data.neo4j.cypher.deprecation        : Neo.ClientNotification.Statement.FeatureDeprecationWarning: This feature is deprecated and will be removed in future versions.

    MATCH path=(powiadomienie:Powiadomienie)-[r1:POWIADAMIA]->(uzyt:Uzytkownik{email: $email})

    WHERE id(powiadomienie) = $id

          ^

    SET powiadomienie.dataUtworzenia =  $time, powiadomienie.avatar = $avatar, r1.przeczytane = false

    RETURN powiadomienie, collect(nodes(path)), collect(relationships(path))

The query used a deprecated function: `id`.

Równanie 10 Log ostrzeżenia

Przykładowy log o błędzie(brak połączenia z bazą danych):

Error starting ApplicationContext. To display the condition evaluation report re-run your application with 'debug' enabled.

2025-01-12T14:22:02.421+01:00 ERROR 28344 --- [yukka] [main] o.s.boot.SpringApplication               : Application run failed

org.springframework.transaction.TransactionSystemException: Could not open a new Neo4j session: Unable to connect to localhost:7687, ensure the database is running and that there is a working network connection to it.

Równanie 11 Log 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, funkcja checkUnban odbanowuje użytkowników, dla których ban się zakończył:

    @Scheduled(cron = "0 0 0 \* \* ?")  // Codziennie

    public void checkUnban() {

        log.info("Sprawdzanie zbanowanych użytkowników");

        List<Uzytkownik> uzytkownicy = uzytkownikRepository.getZbanowaniUzytkownicy();

        if(uzytkownicy.isEmpty()) return;

        for (Uzytkownik uzytkownik : uzytkownicy) {

            if(uzytkownik.getBanDo().isBefore(LocalDate.now())) {

                log.info("Odbanowywanie użytkownika: " + uzytkownik.getNazwa());

                uzytkownikRepository.banUzytkownik(uzytkownik.getNazwa(), false, null, null);

            }

        }

    }

    @EventListener(ApplicationReadyEvent.class)

    public void checkUnbanOnStartup() {

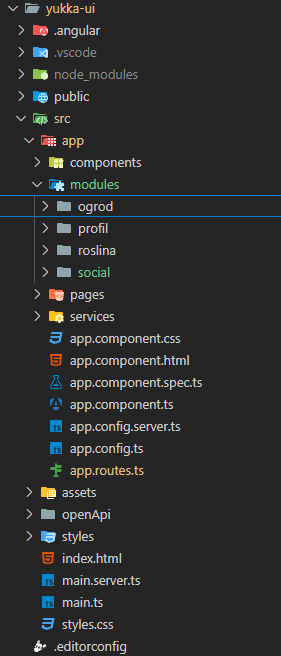
        checkUnban();

    }

Równanie 12 Kod codziennego sprawdzania banów

### Struktura frontendu aplikacji

Frontend jest projektem zrealizowanym w frameworku Angular zawierający strony aplikacji budowanych z pomocą komponentów komunikujących się między sobą głównie poprzez sygnały. Struktura zawiera strony, komponenty je budujące, serwisy do określonych działań, a także moduły grupujące daną część aplikacji w jedno.



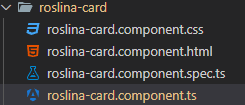
Rysunek 0.21 Struktura frontendu

* Modules – Grupa powiązanych elementów, np. moduł profil zawiera komponenty, serwisy, routy i strony dla profilu użytkownika.
* Components – Niewielkie elementy aplikacji przeznaczone do wielokrotnego użytku, np. karta rośliny.
* Services – Funkcje współdzielone pomiędzy elementami, np. obsługa żądań do backendu, czy weryfikacja parametrów routa..
* Pages – Strony łączące moduły, serwisy i inne części aplikacji.
* App.routes – Routy prowadzące do stron aplikacji.

#### Przykładowy komponent

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

* roslina-card.component.css: definiuje style.
* roslina-card.component.html: struktura html.
* roslina-card.component.spec.ts: Testy jednostkowe.
* roslina-card.component.ts: Logika komponentu (dane, metody, zdarzenia).

Rysunek 0.22 Komponent roslina-card oraz jego widok

Składa się on z plików:

Zawartość pliku roslina-card.component.ts:

import { Component, Input } from '@angular/core';

import { RoslinaResponse } from '../../../../services/models';

import { RouterModule } from '@angular/router';

import { CommonModule } from '@angular/common';

@Component({

  selector: 'app-roslina-card',

  standalone: true,

  imports: [CommonModule, RouterModule],

  templateUrl: './roslina-card.component.html',

  styleUrl: './roslina-card.component.css'

})

export class RoslinaCardComponent {

  @Input()  roslina:RoslinaResponse = {};

  private \_roslinaObraz: string | undefined;

  getRoslina(): RoslinaResponse {

    return this.roslina;

  }

  setRoslina(roslina: RoslinaResponse) {

    this.roslina = roslina;

  }

  getRoslinaObraz(): string | undefined {

    if(this.roslina.obraz) {

      return 'data:image/jpeg;base64,' + this.roslina.obraz;

    }

    return this.\_roslinaObraz;

  }

}

Równanie 13 Kod roslina-card.component.ts

Plik roslina-card.component.ts zawiera importy potrzebnych modułów, klasę komponentu z obiektem wejściowym @Input dla rośliny, getter, setter i metodę zwracająca obraz rośliny w formacie base64.

Zawartość pliku roslina-card.component.html:

<div class="card">

  <img [src]="getRoslinaObraz()" class="card-img-top" alt="Card Image">

  <div class="card-body d-flex flex-column" style="background: var(--bs-secondary);border-color: var(--bs-primary-text-emphasis);">

    <p \*ngIf="!roslina.roslinaUzytkownika" class="text-truncate mb-1 text-white-50 fs-6 card-text mb-0">({{roslina.nazwaLacinska}})</p>

    <h5 class="card-title">{{roslina.nazwa}}</h5>

    <p class=" limit-to-3-lines">{{roslina.opis}}</p>

    <a [routerLink]="['/rosliny', roslina.uuid]" class="btn btn-primary mt-auto align-self-start">Przejdź do rośliny</a>

  </div>

</div>

Równanie 14 Kod roslina-card.component.html

W podanym pliku roslina-card.component.html wywołuje się metodę getRoslinaObraz() zwracająca obraz rośliny do tagu <img> , a także wypisywane są właściwości rośliny z pominięciem nazwy łacińskiej, jeśli jest to roślina użytkownika (\*ngIf).

Komponent ten jest używany do pokazywania wyników z listy roślin. Korzystając z pętli ngFor, można dla każdej rośliny w liście utworzyć komponent roslina-card, przekazując do niego dane konkretnej rośliny:

<app-roslina-card

            \*ngFor="let roslina of roslinaResponse?.content"

            [roslina]="roslina">

</app-roslina-card>

Równanie 15 Przykładowa pętla tworząca karty roślin

#### Przykład metody API z serwisu

Metody API zostały wygenerowane przy użyciu narzędzia ng-openapi-gen. Są one podzielone na metody zwracające cały HttpResponse i metody zwracające jedynie ciało. Każda metoda przyjmuje odpowiednie parametry oraz opcjonalnie HttpContext. Poniżej znajduje się przykładowa metoda API w pliku authentication.service.ts:

  static readonly LoginPath = '/api/auth/login';

  login$Response(params: Login$Params, context?: HttpContext): Observable<StrictHttpResponse<AuthenticationResponse>> {

    return login(this.http, this.rootUrl, params, context);

  }

  login(params: Login$Params, context?: HttpContext): Observable<AuthenticationResponse> {

    return this.login$Response(params, context).pipe(

      map((r: StrictHttpResponse<AuthenticationResponse>): AuthenticationResponse => r.body)

    );

  }

Równanie 16 Fragment kodu z authentication.service.ts

Korzysta się tutaj dodatkowo z wygenerowanej funkcji login osadzonej w osobnym pliku login.ts, gdzie wykonuje żądanie logowania użytkownika zgodnie z dokumentacją OpenApi, przyjmując parametry i zwracając odpowiedź.

export function login(http: HttpClient, rootUrl: string, params: Login$Params, context?: HttpContext): Observable<StrictHttpResponse<AuthenticationResponse>> {

  const rb = new RequestBuilder(rootUrl, login.PATH, 'post');

  if (params) {

    rb.body(params.body, 'application/json');

  }

  return http.request(

    rb.build({ responseType: 'json', accept: 'application/json', context })

  ).pipe(

    filter((r: any): r is HttpResponse<any> => r instanceof HttpResponse),

    map((r: HttpResponse<any>) => {

      return r as StrictHttpResponse<AuthenticationResponse>;

    })

  );

}

login.PATH = '/api/auth/login';

Równanie 17 funkcja login w login.ts

#### Seedery

Aplikacja posiada seedery do poszczególnych części aplikacji.

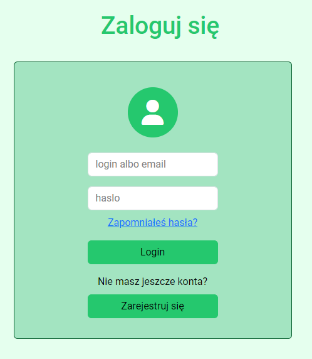
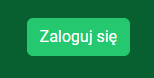
* RoslinaImporterService – Importuje rośliny z podanego pliku .csv w ilości roslina.seed.amount. Ze względu na ilość i złożoność roślin, zamiast usuwania kolejno każdego węzła i relacji, czyszczona jest cała baza danych.
* GeneralSeederService – Importuje użytkowników, posty, komentarze, rozmowy prywatne, rośliny użytkownika i rośliny na działce.
* SocialSeeder – Seeduje część społecznościową (posty, komentarze, odpowiedzi) w ramach GeneralSeederService, z możliwością dostosowania liczby postów, komentarzy i głębokości odpowiedzi.
* RoslinaWlasnaSeeder – Seeduje rośliny użytkownika. Używane w GeneralSeederService.

## Opis funkcjonalności systemu

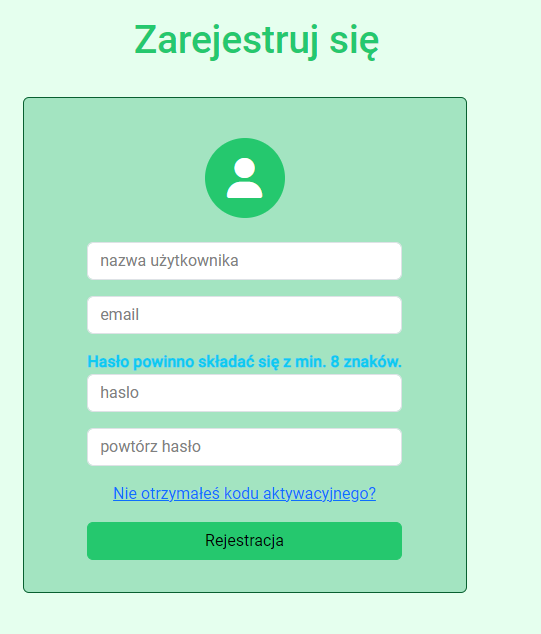
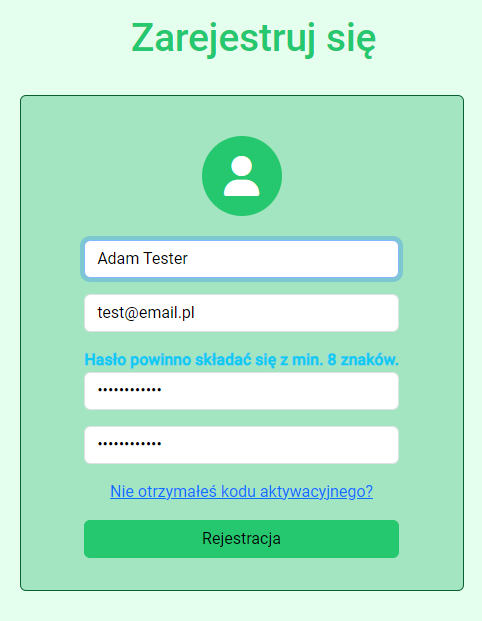
### Funkcjonalności zwykłego użytkownika

#### Rejestracja

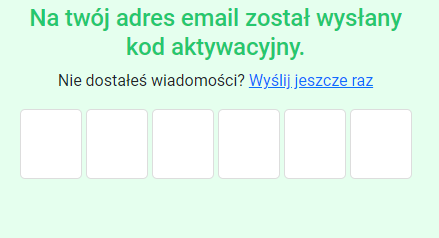
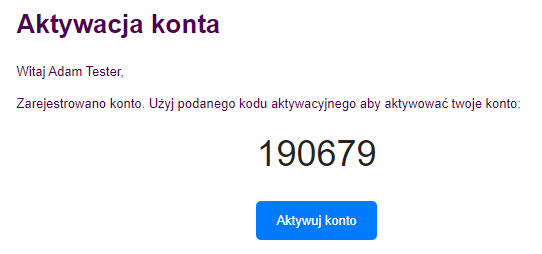
Aby się zarejestrować, należy w pasku nawigacyjnym wcisnąć przycisk „Zaloguj się” przenoszący do strony logowania. Następnie w panelu logowania wybrać opcję „Zarejestruj się” i wprowadzić wymagane dane (przykładowe dane poniżej). Po rejestracji użytkownik zostaje przeniesiony na stronę aktywacji konta, gdzie wprowadza kod otrzymany na adres e-mail.



Rysunek 0.23 Przycisk „Zaloguj się” i panel logowania

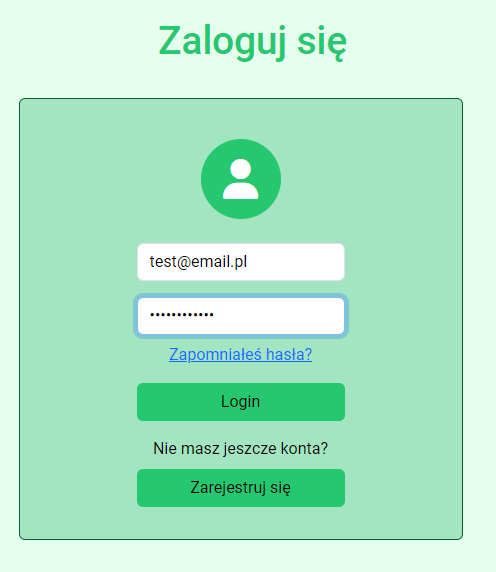
 

Rysunek 0.24 Panel rejestracji

Rysunek 0.25 Panel aktywacji i kod z wiadomości email

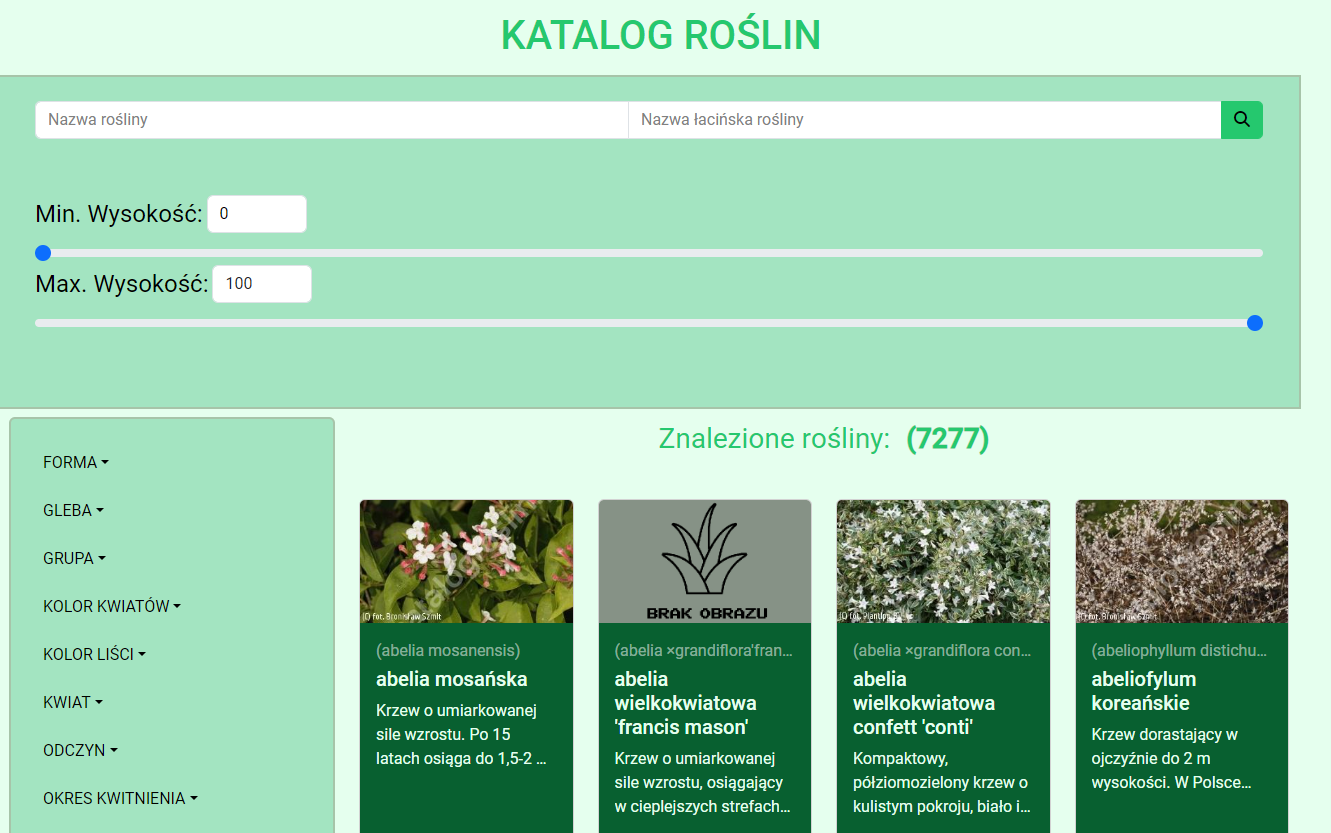
Aby się zalogować, należy w panelu logowania podać adres email lub nazwę użytkownika oraz jego hasło. Po kliknięciu przycisku „Login” użytkownik zostanie przeniesiony do strony głównej, tym razem zalogowany na swoje konto:



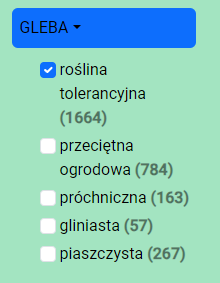
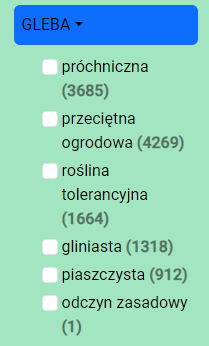
Rysunek 0.28 Panel logowania z danymi i widok po zalogowaniu się

#### Przeglądanie roślin w katalogu

Aby przeglądać rośliny w katalogu, należy wybrać „Przeglądaj katalog” w pasku nawigacyjnym, przenoszący na stronę katalogu roślin. W katalogu można wyszukiwać rośliny według nazwy, wysokości, cech (np. rodzaj gleby, miesiące kwitnienia) oraz wprowadzać dodatkowe filtry. Użytkownik może łatwo usunąć wybrane cechy klikając znak „X”. Katalog obsługuje paginację, umożliwiając przechodzenie między stronami.



Rysunek 0.31 Widok katalogu roślin



Rysunek 0.33 Lista cech typu Gleba przed i po wyszukaniu



Rysunek 0.35 Widok tagów cech

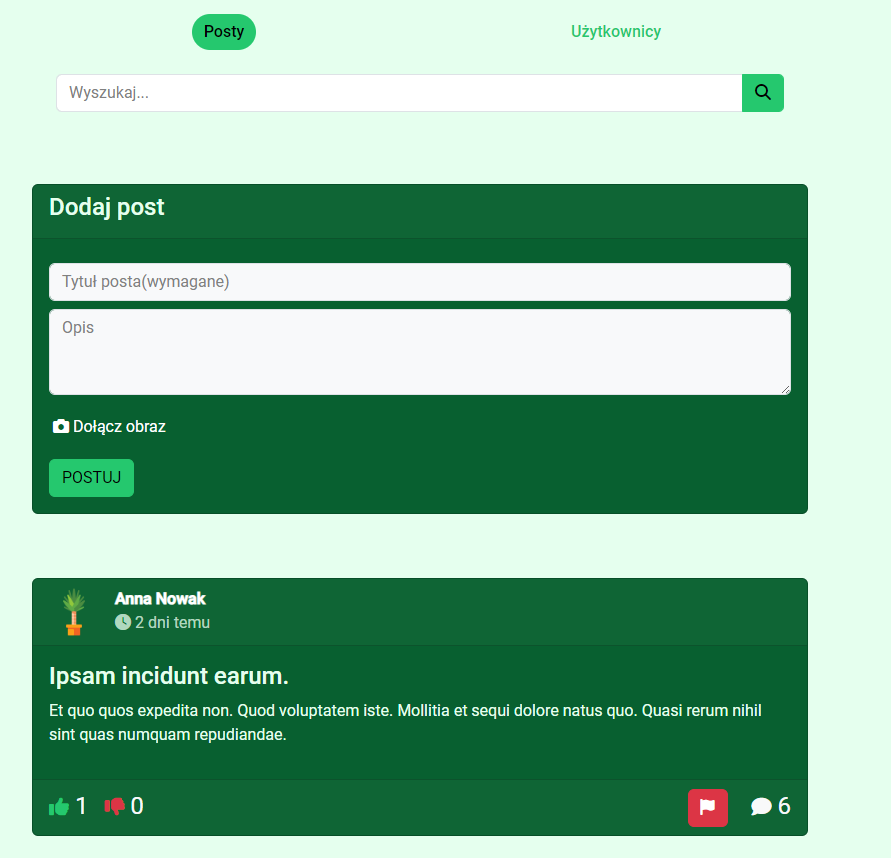
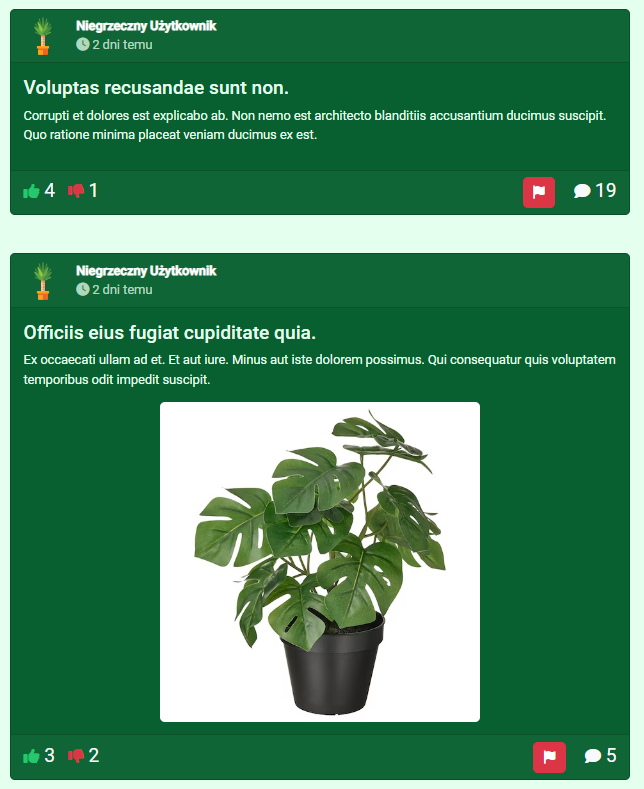
Aby zobaczyć roślinę z katalogu, należy kliknąć przycisk „Przejdź do rośliny”, który przeniesie użytkownika do strony rośliny. Na stronie rośliny znajduje się jej nazwa, nazwa łacińska, obraz (jeśli ma), opis (jeśli ma), oraz wszystkie jej cechy:



Rysunek 0.37 Widok strony rośliny

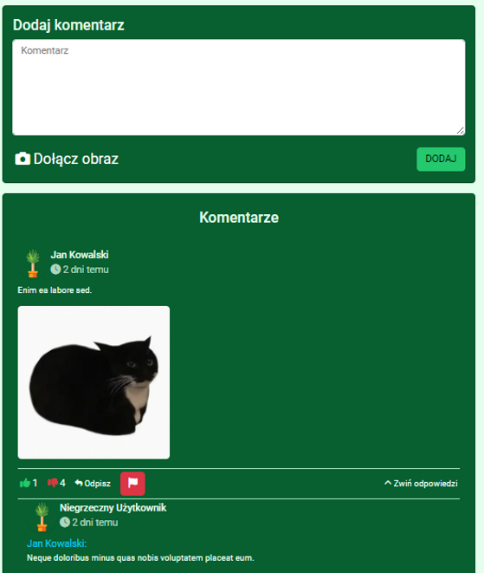
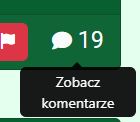
#### Przeglądanie postów i komentarzy

Aby przejść do postów, należy wybrać „Społeczność” w pasku nawigacyjnym. Panel społeczności ma wyszukiwarkę postów i użytkowników. W postach zastosowano mechanizm „Infinite Scrolling”, ładujący nowe posty po dotarciu do końca strony. Posty mają autora, tytuł, opis, obraz (opcjonalnie), ocenę oraz liczbę komentarzy. Jest też przycisk zgłaszania postu.

Rysunek 0.39 Wyszukiwarka postów i przykłady postów

Aby przejść do komentarzy, wystarczy kliknąć na liczbę komentarzy lub ikonkę dymku. Strona posta umożliwia dodanie komentarza oraz wyświetla listę wszystkich komentarzy. Komentarze mają opcje oceniania, odpowiadania i zwijania zagnieżdżonych odpowiedzi.

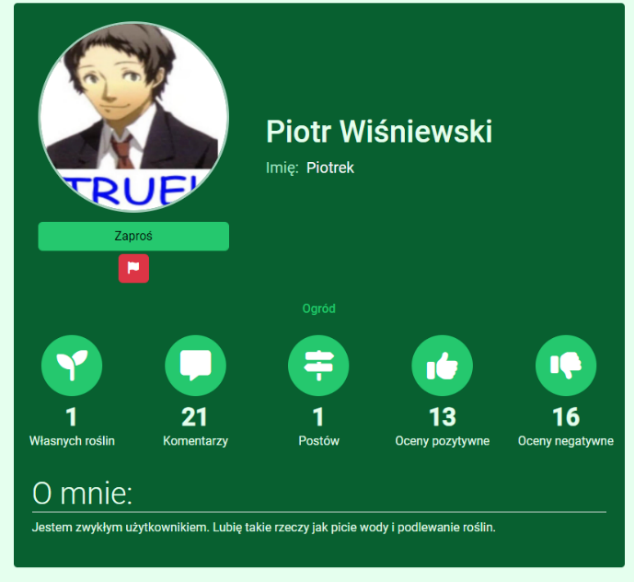
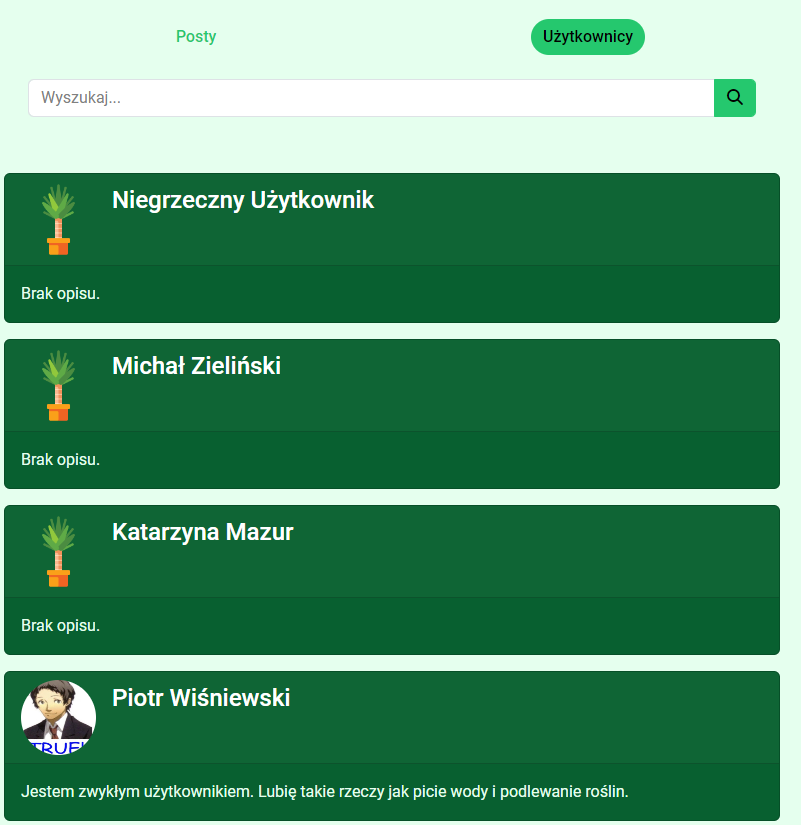


Rysunek 0.42 Widok komentarzy postu

Komentarz posiada oceny, możliwość odpisania, zgłodzenia, a także zwinięcia odpowiedzi jeśli takie posiada. Dla postu, komentarze mają rozwinięte odpowiedzi, jeśli nie są one zbyt zagnieżdżone (odpowiedź na odpowiedź na odpowiedź…).

#### Przeglądanie użytkowników

W panelu „Społeczność” należy kliknąć na przycisk „Użytkownicy”, który zabierze użytkownika do listy użytkowników:



Rysunek 0.43 Lista użytkowników i profil jednego z nich

#### Przeglądanie ogrodów

Aby przeglądać ogrody, zalogowany użytkownik musi przejść do ogrodu klikając „Mój ogród” w pasku nawigacyjnym, lub wybierając opcję „Ogród” na profilu, jeśli jest on publiczny. Po przejściu do ogrodu użytkownik zobaczy nazwę ogrodu, listę działek z liczbą roślin oraz przejście do katalogu własnych roślin. Aby przejść do działki, należy kliknąć „Zobacz” na wybranej działce.



Rysunek 0.50 Widok ogrodu użytkownika

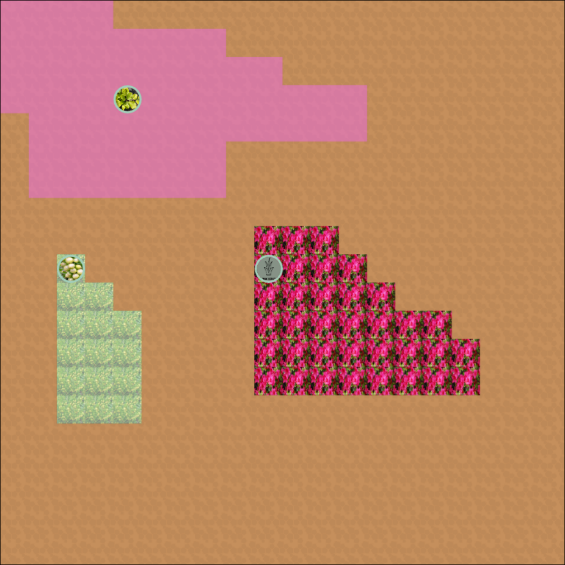
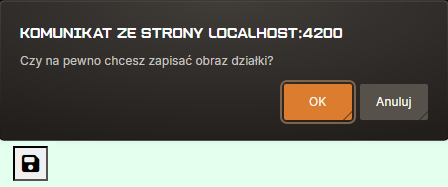
Strona działki pokazuje nazwę działki, rośliny w formie obrazów i kafelków. Istnieje możliwość wyboru różnych trybów wyświetlania roślin, takich jak kolorowanie kafelków, wyświetlanie samej tekstury roślin lub oba tryby jednocześnie. Użytkownik może przybliżać/oddalać widok działki oraz przemieszczać się po niej poprzez przeciąganie.

Strona działki zawiera nazwę działki, rośliny pokazane w postaci ich obrazów oraz kafelków do nich należących w różnych trybach wyświetlania(kolorowanie kafelków, wyświetlanie samej tekstury, lub oba), możliwość przybliżania/oddalania widoku, przemieszczanie się przez działkę poprzez przeciąganie, a także możliwość dokładniejszego zobaczenia rośliny w trybie wybierania.



Rysunek 0.51 Widok działki

Dodatkowo, można zapisać obraz działki klikając w ikonę floppy-disk i potwierdzając powiadomienie:



Rysunek 0.53 Potwierdzenie i stworzony obraz działki

### Funkcjonalności pracownika(TODO)

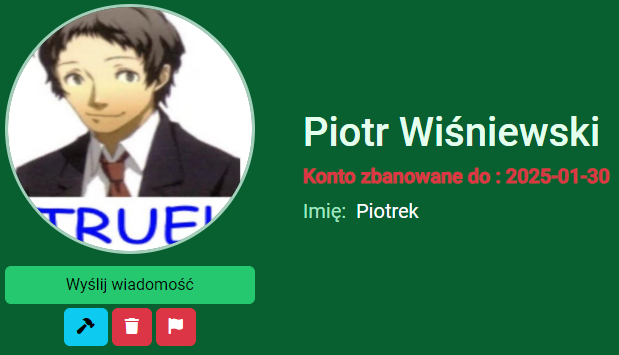
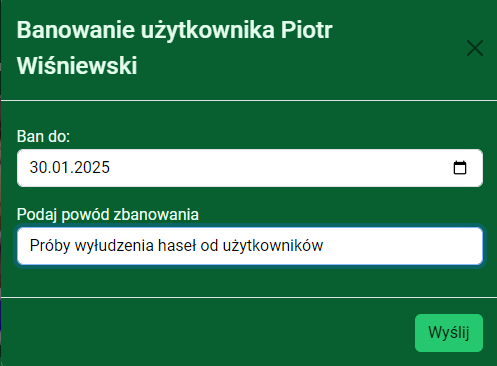
Pracownik dziedziczy wszystkie funkcjonalności użytkownika. Oprócz tego posiada swoje własne.

#### Banowanie użytkowników

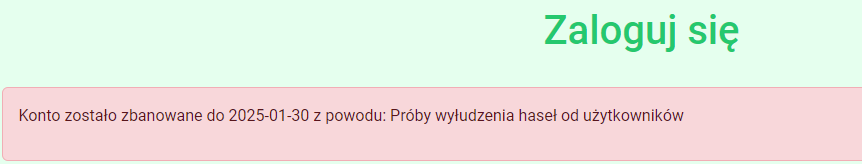
Pracownicy mogą banować użytkowników przechodząc do ich profilu i klikając ikonę młotka, przez co pojawi się okno banowania użytkownika, gdzie trzeba podać datę zakończenia oraz powód. Konto użytkownika zostanie automatycznie odbanowane po tej dacie. Ewentualnie, pracownik może odbanować go ręcznie klikając jeszcze raz na ikonę młotka i potwierdzając powiadomienie.



Rysunek 0.45 Opcja banowania użytkownika



Rysunek 0.46 Formularz banowania i użytkownik po zbanowaniu



Rysunek 0.47 Komunikat o zbanowaniu przy próbie zalogowania się

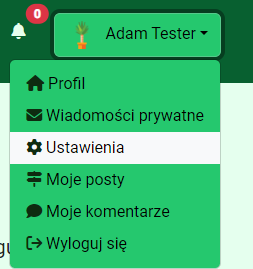
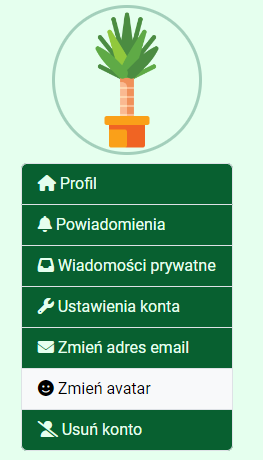
### Funkcjonalności admina(TODO)

### Zarządzanie zasobami

#### Zarządzanie kontem

##### Zmiana avatara

Aby zmienić avatar użytkownika, należy w opcji użytkownika przejść do ustawień, a następnie wybrać opcję „Zmień avatar”:

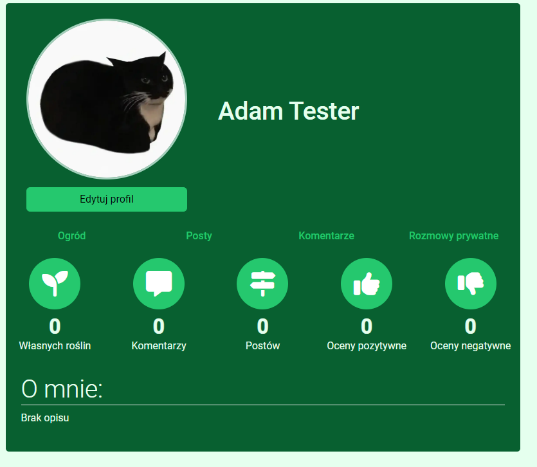
 

Rysunek 0.54 Przejście do ustawień i do zmiany avatara



Rysunek 0.55 Panel zmiany avatara

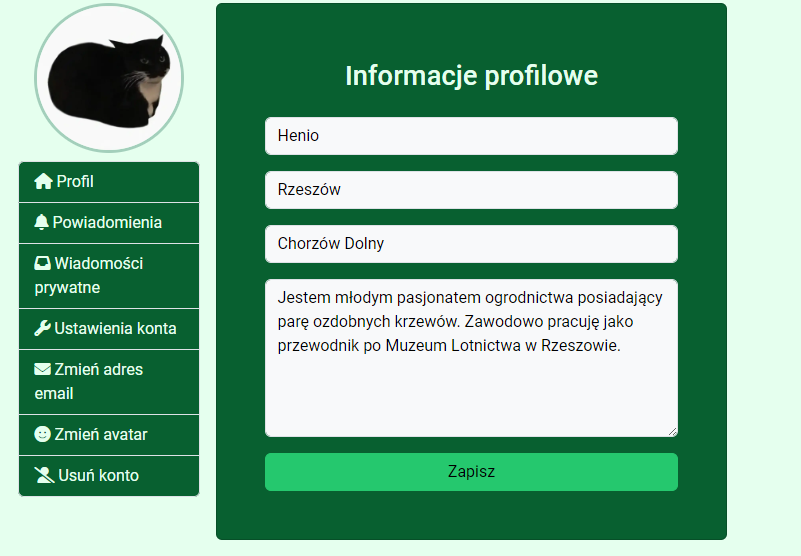
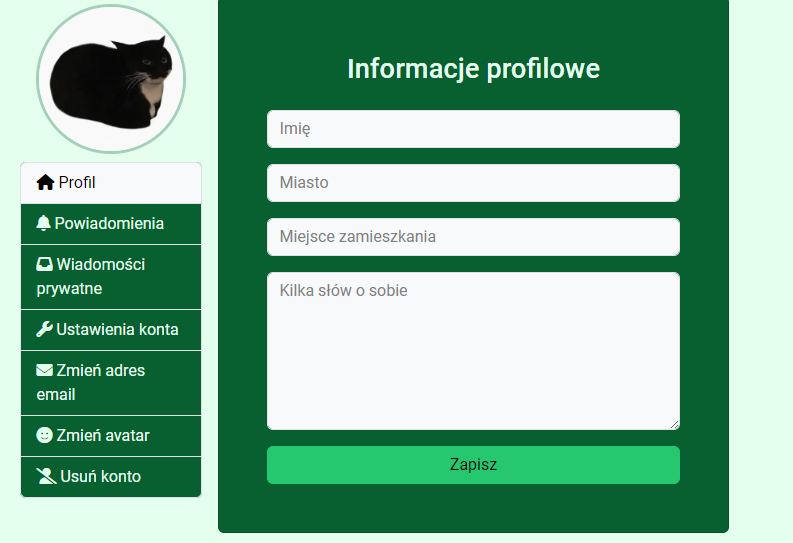
Aby zmienić avatar, należy dołączyć obraz. Po kliknięciu przycisku „Dodaj obraz” avatar użytkownika zostanie zmieniony na nowy i zostanie on przeniesiony z powrotem do swojego profilu:



Rysunek 0.56 Profil po zmianie avatara

##### Edycja profilu

W panelu ustawień należy kliknąć opcję „Profil”, która przeniesie użytkownika do widoku edycji danych profilowych. Można edytować dane profilowe, podając lub usuwając imię, miasto, miejsce zamieszkania oraz opis.



Rysunek 0.57 Panel edycji profilu przed i po podaniu danych

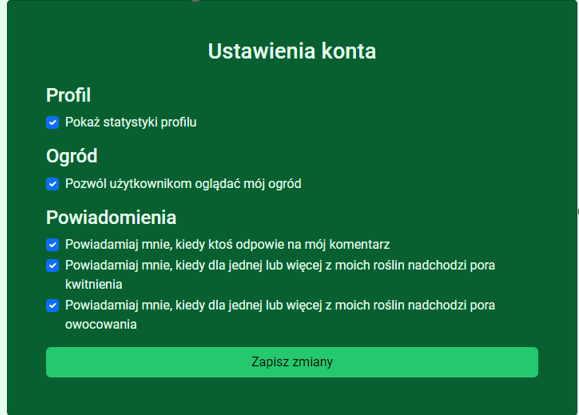
Po wciśnięciu przycisku „Zapisz” wyskoczy komunikat o pomyślnej aktualizacji profilu



Rysunek 0.58 Profil po edycji

##### Zmiana ustawień konta

Aby zmienić ustawienia konta, należy w panelu ustawień przejść do opcji „Ustawienia konta”:

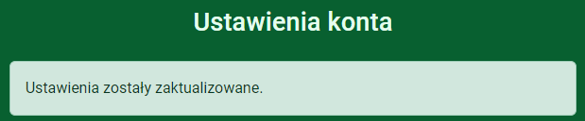


Rysunek 0.59 Panel ustawień konta

W panelu konta można w skrócie ustawić następujące opcje:

* Pokazywanie statystyk profilu
* Ukryć ogród przed innymi użytkownikami (nie tyczy się pracowników/adminów)
* Dostosować, o jakich rzeczach aplikacja powinna powiadamiać użytkownika

Po zmianie ustawień i kliknięciu „Zapisz zmiany” wyskoczy komunikat o pomyślnej operacji. Można teraz przejść do profilu i zobaczyć, że statystyki użytkownika oraz panel „Ogród” zostały ukryte.

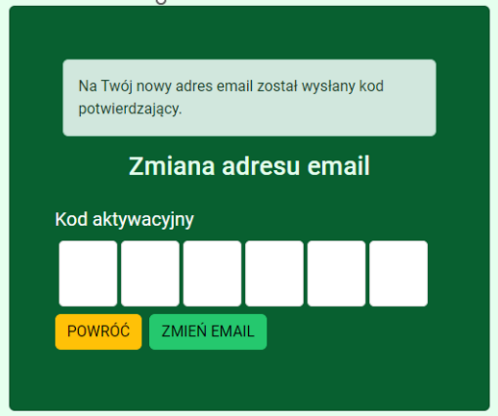
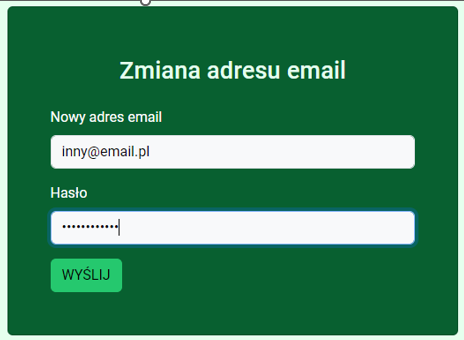




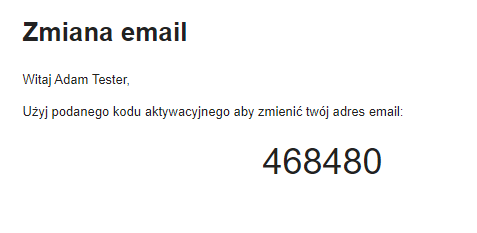
Rysunek 0.60 Komunikat o zmianie ustawień i profil po zmianie ustawień konta

##### Zmiana adresu email

Aby zmienić adres email, należy w ustawieniach przejść do opcji „Zmień adres email”, gdzie trzeba w panelu podać nowy adres email oraz obecne hasło. Po kliknięciu przycisku „Wyślij” zostanie wysłana wiadomość na nowy adres email, gdzie trzeba będzie podać wysłany kod aktywacyjny.

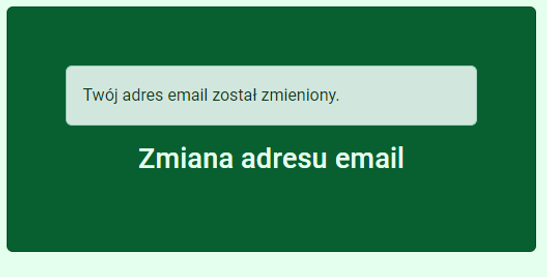


Rysunek 0.61 Panel zmiany adresu email



Rysunek 0.62 Kod aktywacyjny

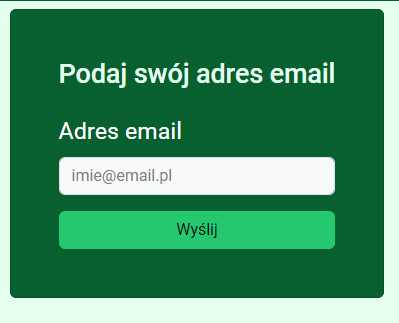
Po podaniu kodu i kliknięciu przycisku „Zmień email” pokaże się komunikat o zakończeniu zmiany adresu email.



Rysunek 0.63 Panel zmiany email po podaniu kodu

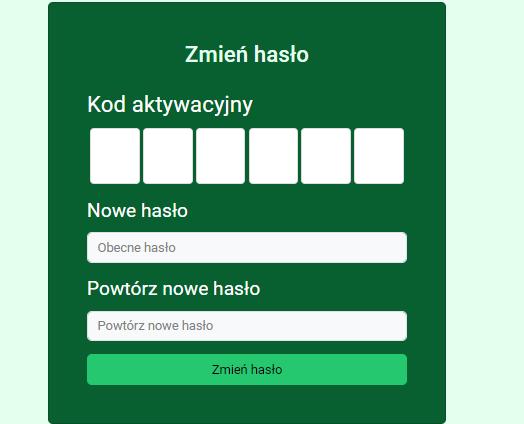
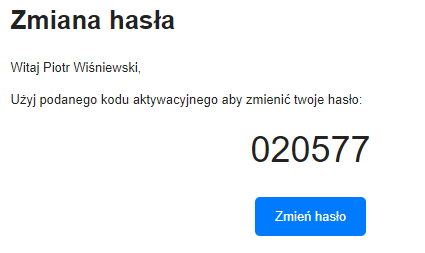
##### Zmiana hasła

Aby zmienić hasło, należy w panelu logowania kliknąć link „Zapomniałeś hasła?”, który przeniesie użytkownika do strony z resetowaniem hasła. Tam należy podać adres email użytkownika, na który zostanie wysłana wiadomość o zmianie hasła:

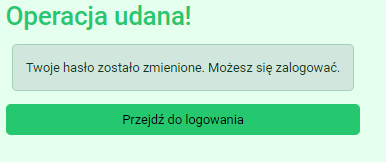


Rysunek 0.64 Link do zmiany hasła i formularz

Po wysłaniu wiadomości trzeba będzie podać w następnej części kod oraz nowe hasło, gdzie po poprawnym podaniu hasło zostanie zmienione:



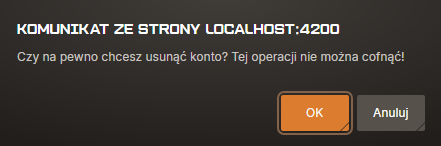
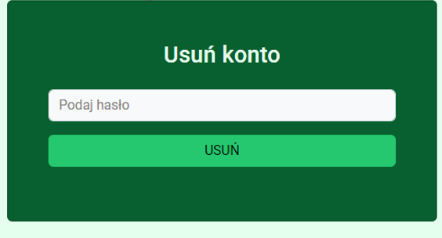
Rysunek 0.65 Formularz zmiany hasła



Rysunek 0.66 Udana zmiana hasła

##### Usuwanie konta

Aby usunąć konto, należy w ustawieniach przejść do panelu „Usuń konto”, a następnie podać hasło i kliknąć przycisk „Usuń”. Po naciśnięciu przycisku wyskoczy ostrzeżenie, po którego potwierdzeniu aplikacja przystąpi do usuwania konta, wylogowaniu użytkownika i przeniesieniu go z powrotem do strony głównej.

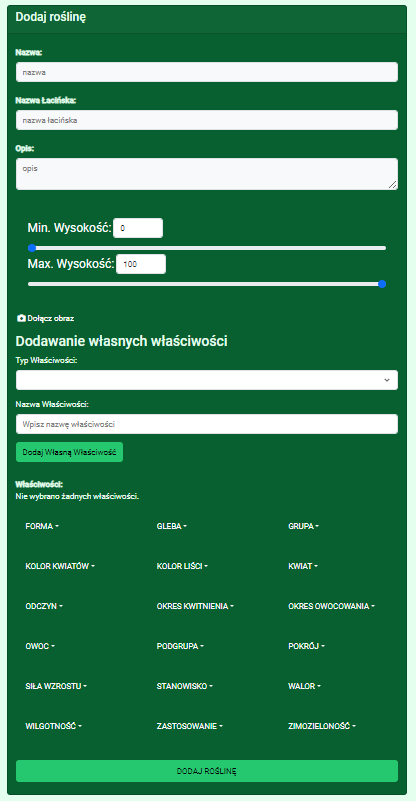


Rysunek 0.67 Panel usuwania konta i potwierdzenie

#### Zarządzanie roślinami

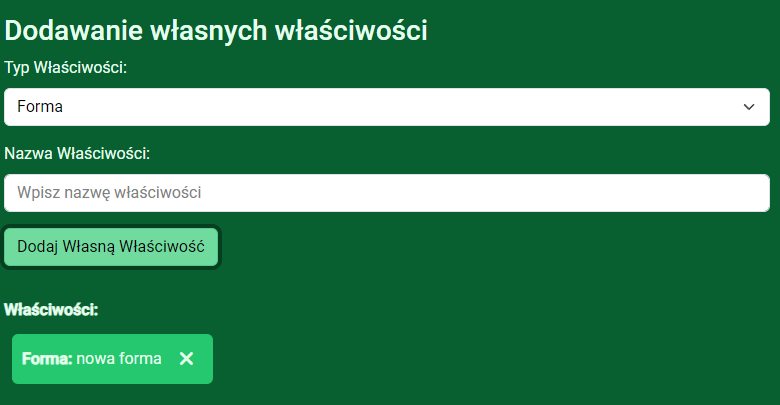
##### Dodawanie rośliny

Aby dodać roślinę do katalogu, należy jako pracownik w katalogu roślin kliknąć przycisk „Dodaj roślinę” przenoszący użytkownika do strony dodawania. Jeśli zamiast tego chce się dodać własną roślinę, należy przejść do strony „Mój ogród”, wybrać przeglądanie swoich roślin i tam kliknąć przycisk „Dodaj roślinę”.



Rysunek 0.68 Przycisk dodawania rośliny i formularz

Na stronie dodawania rośliny należy wpisać parametry rośliny, opcjonalnie dołączyć obraz i przypisać wybrane rodzaje cech. Można również stworzyć jedną z cech poprzez wybranie jej typu i podanie własnej nazwy:



Rysunek 0.69 Nowa właściwość rośliny

Po podaniu danych należy kliknąć przycisk zatwierdzenia. Przy poprawnym dodaniu rośliny do katalogu zostanie się przeniesionym do jej strony:

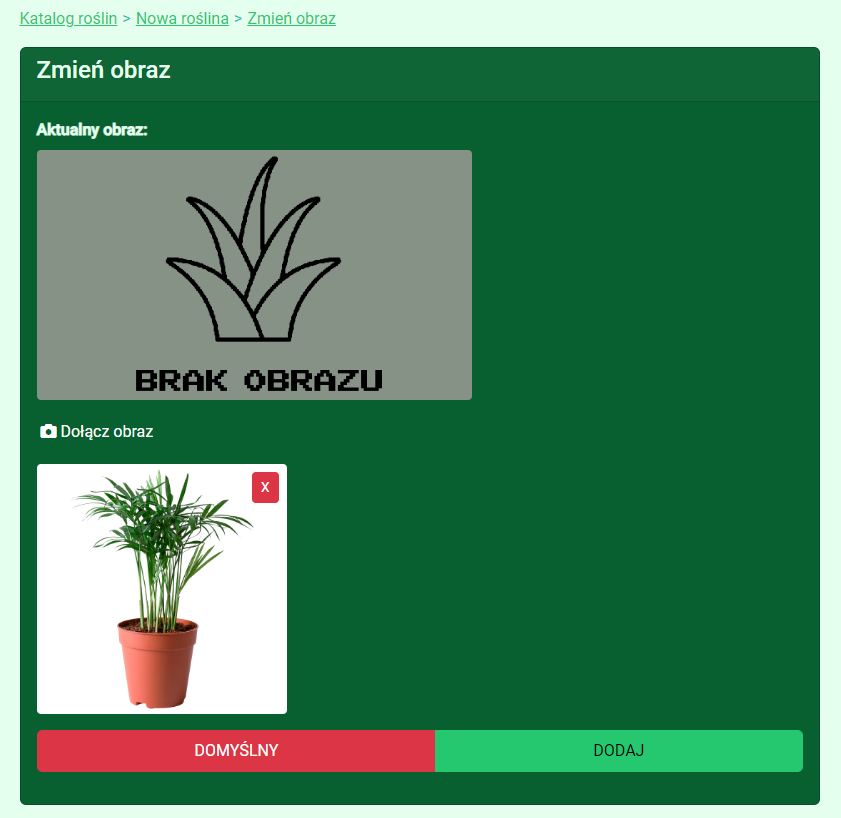


Rysunek 0.70 Dodana roślina

##### Aktualizacja rośliny

Aby aktualizować roślinę, należy jako pracownik (jeśli jest to roślina z katalogu) albo użytkownik (roślina własna) na jej stronie kliknąć przycisk „Edytuj”, który przeniesie użytkownika do strony edycji rośliny będący taki sam, jak strona dodawania rośliny z tym wyjątkiem, że nie podaje się tam obrazu. Po zatwierdzeniu zmian zostanie się z powrotem przeniesionym do strony rośliny.

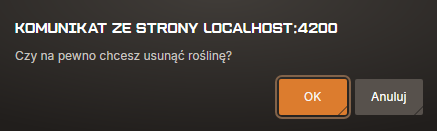
Aby zmienić obraz rośliny, trzeba na stronie rośliny kliknąć przycisk „Zmień obraz”, który przeniesie użytkownika do strony zmiany obrazu, gdzie można ustawić nowy obraz, albo przywrócić domyślny obraz dla rośliny. Po wybraniu jednej z tych opcji obraz rośliny zostanie zaktualizowany.

Rysunek 0.71 Zmiana obrazu rośliny

##### Usuwanie rośliny

Aby usunąć roślinę, należy na jej stronie kliknąć przycisk „usuń” i zatwierdzić. Po tej operacji zostanie się przeniesionym do katalogu:



Rysunek 0.72 Potwierdzenie usuwania rośliny

#### Zarządzanie własnymi roślinami

Aby zarządzać własnymi roślinami, należy w pasku nawigacyjnym wybrać opcję „Mój ogród”, a następnie w części „Rośliny użytkownika” kliknąć przycisk „Zobacz”, który przeniesie użytkownika do jego katalogu z roślinami. Tam można takie same operacje, jak w **Błąd! Nie można odnaleźć źródła odwołania.** lecz są one wykonywane na własnych roślinach, zamiast na roślinach w oficjalnym katalogu.

#### Zarządzanie postami i komentarzami

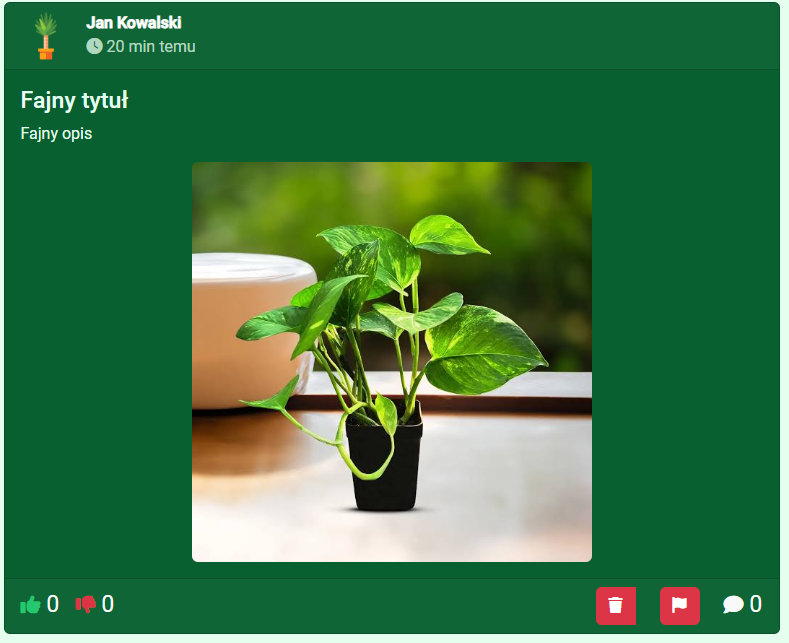
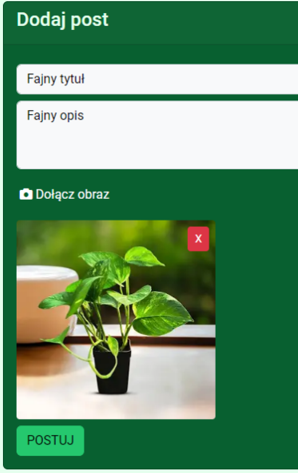
Swoje posty i komentarze można przeglądać, klikając w panelu użytkownika opcję „Moje posty” lub „Moje komentarze”, które przekieruje użytkownika do strony z jego postami/komentarzami:



Rysunek 0.73 Przejście do postów użytkownika

##### Dodanie postu

Post można dodać w panelu „Społeczność”, gdzie trzeba podać tytuł i opcjonalnie opis oraz obraz postu. Po zatwierdzeniu zostanie się przeniesionym do nowego postu. Post można dodawać co 30 sekund.



Rysunek 0.74 Formularz dodania posta i post po dodaniu

##### Usuwanie postu

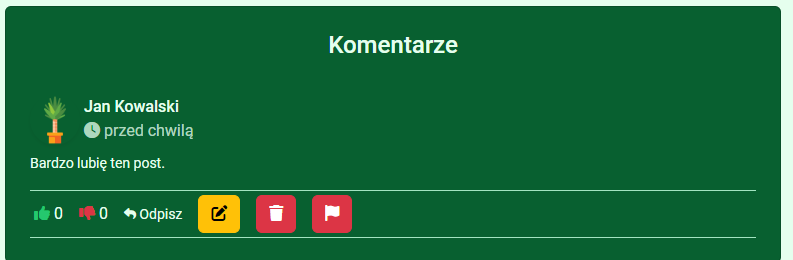
Aby usunąć post, należy kliknąć ikonę śmietnika i zatwierdzić usuwanie. Po zatwierdzeniu, post wraz z komentarzami i ocenami zostanie usunięty, a użytkownik zostanie przekierowany z powrotem do strony społeczności.



Rysunek 0.75 Usuwanie posta

##### Dodanie komentarza

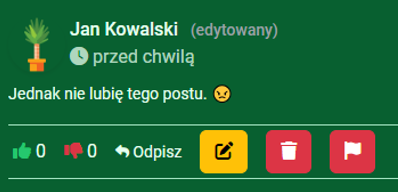
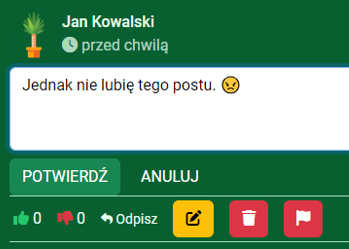
Aby dodać komentarz, należy na stronie postu wypełnić formularz, a następnie zatwierdzić. Komentarz można dodawać co 30 sekund.



Rysunek 0.76 Dodawanie komentarza i dodany komentarz

##### Edycja komentarza

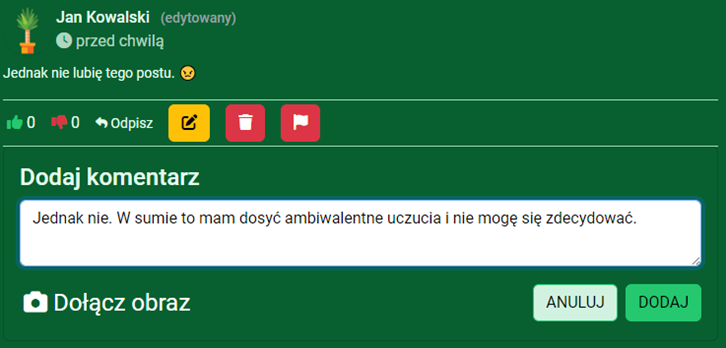
Swój komentarz można edytować, klikając żółtą ikonę, która pokaże formularz edycji, po którego zatwierdzeniu komentarz zostanie zmieniony:

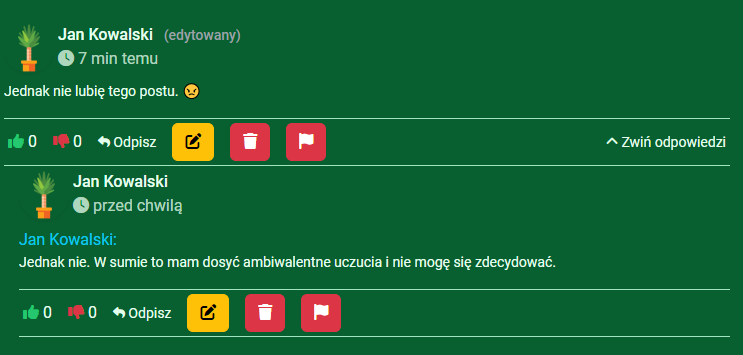


Rysunek 0.77 Edycja komentarza i komentarz po edycji

##### Dodanie odpowiedzi

Aby dodać odpowiedź do komentarza, należy kliknąć „Odpisz” w komentarzu, a następnie wypełnić formularz jak do komentarza:





Rysunek 0.78 Dodawanie odpowiedzi i odpowiedź po edycji

##### Ocena postu lub komentarza

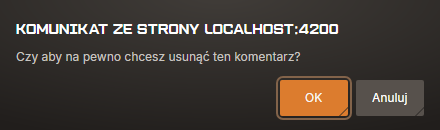
Komentarz i post można ocenić, jako zalogowany użytkownik, klikając na kciuk w górę lub kciuk w dół. Nie można oceniać własnych komentarzy:



Rysunek 0.79 Ocena postu/komentarza

##### Usuwanie komentarza

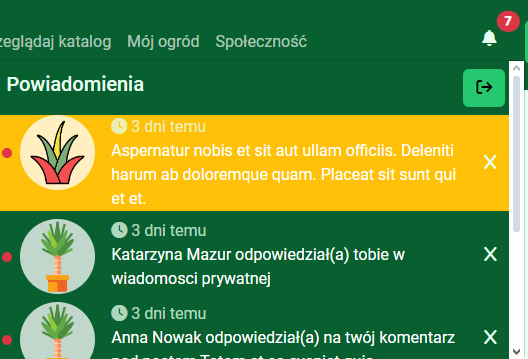
Aby usunąć komentarz, należy w komentarzy kliknąć ikonę śmietnika i potwierdzić. Wraz z komentarzem zostaną oceny i odpowiedzi do niego:



Rysunek 0.80 Ikona usuwania

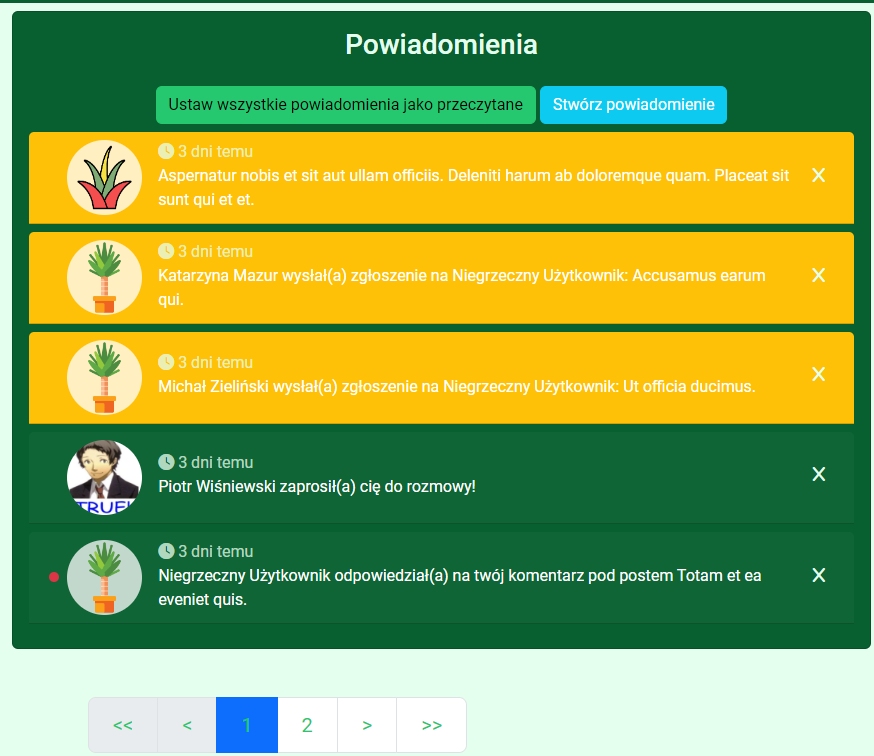
#### Zarządzanie powiadomieniami

Aby zarządzać powiadomieniami, należy na pasku nawigacyjnym kliknąć w ikonę dzwonka pokazujący liczbę nieprzeczytanych powiadomień. Ukaże się okno z dynamicznie ładowaną listą powiadomień, które zostają przeczytane przy najechaniu na nie myszki i które można usuwać, klikając X po prawej. Dodatkowo, zgłoszenia i powiadomienia specjalne są oznaczone żółtym kolorem.



Rysunek 0.81 Okno powiadomień

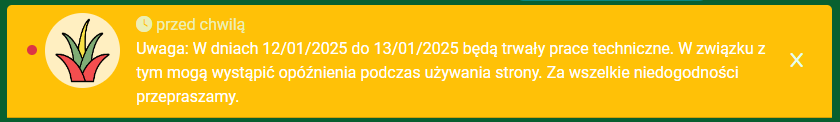
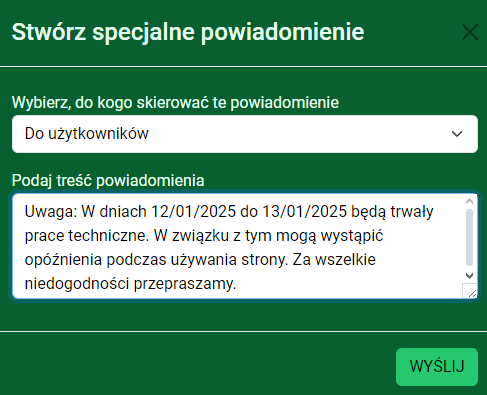
Można również przejść do strony z powiadomieniami klikając przycisk w prawym górnym rogu, który zabierze użytkownika do strony z powiadomieniami, gdzie są pokazywane w formie paginacji. Na tej stronie można ustawić wszystkie powiadomienia jako przeczytane, albo stworzyć powiadomienie specjalne, jeśli jest się pracownikiem albo adminem.



Rysunek 0.82 Panel powiadomień pracownika

##### Tworzenie specjalnego powiadomienia

Aby stworzyć specjalne powiadomienie, należy jako pracownik 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).



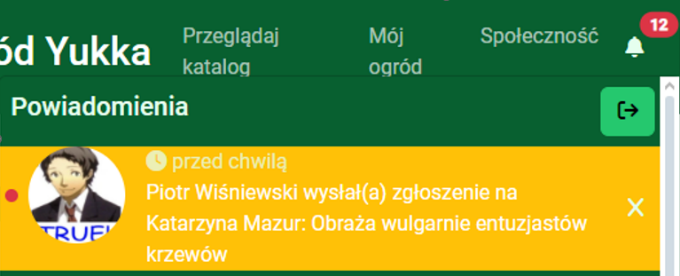
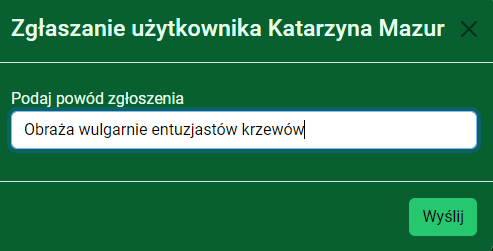
Rysunek 0.83 Okno specjalnego powiadomienia i stworzone powiadomienie

##### Dodawanie zgłoszeń

Zgłoszenie można dodać, klikając flagę przy poście, komentarzu, czy użytkowniku na profilu. Po kliknięciu wyskoczy okno z powodem zgłoszenia. Po potwierdzeniu zgłoszenie zostanie wysłane do jednego z pracowników:



Rysunek 0.84 Przycisk zgłaszania



Rysunek 0.85 Opis zgłoszenia i widok wysłanego zgłoszenia

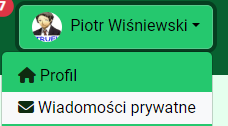
##### Usuwanie powiadomienia

Powiadomienie można usunąć, klikając przycisk „X” po prawej od niego. Warto jednak zauważyć, że nie usuwa się powiadomienia, lecz relację łączącą użytkownika z powiadomieniem, a samo powiadomienie jest usuwane, jeśli już nikt nie ma z nim relacji.

Uwaga: Zgłoszeń się nie usuwa. Są one zamiast tego ukrywane, gdyż zgłoszenia są przypisywane do pracowników posiadających ich najmniej.

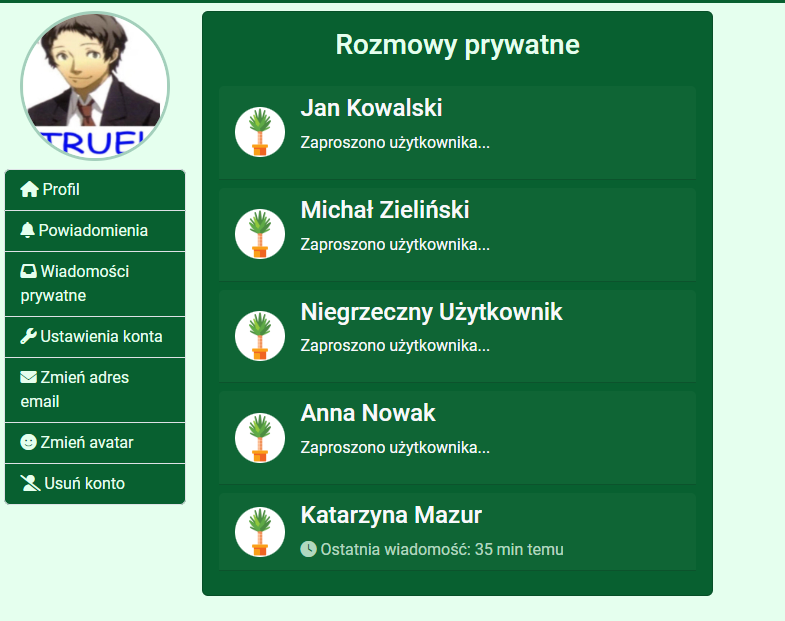
#### Zarządzanie rozmowami

Aby przeglądać rozmowy, trzeba być zalogowanym. Można przejść do rozmów poprzez wybranie opcji „Wiadomości prywatne”:



Rysunek 0.86 Przejście do rozmów

W panelu rozmów prywatnych są rozmowy użytkownika z innymi użytkownikami, a także oczekujące zaproszenia od/do.



Rysunek 0.87 Rozmowy Prywatne

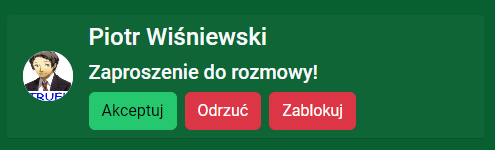
##### Wysyłanie zaproszenia

Aby wysłać zaproszenie, trzeba na profilu użytkownika wcisnąć „Zaproś”:



Rysunek 0.88 Przycisk "Zaproś"

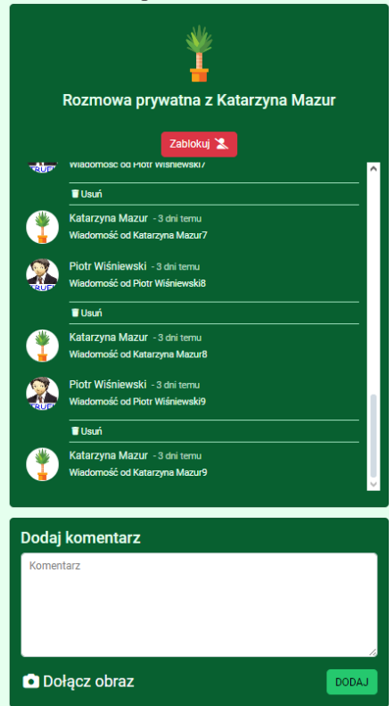
Zaproszenie użytkownika można zaakceptować, odrzucić, lub zablokować użytkownika. Jeśli zaakceptuje się zaproszenie, użytkownicy mogą wysyłać do siebie wiadomości.



Rysunek 0.89 Przykładowe zaproszenie do rozmowy

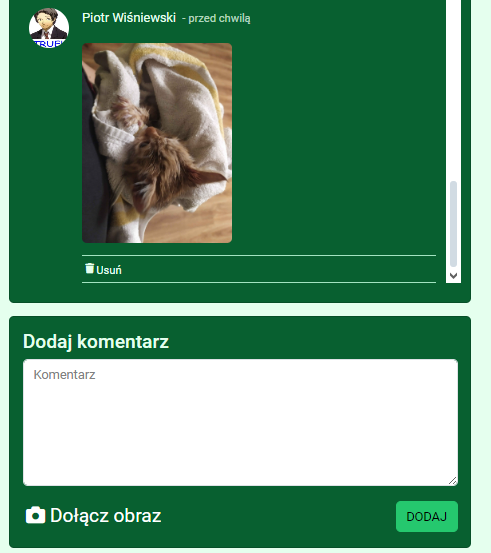
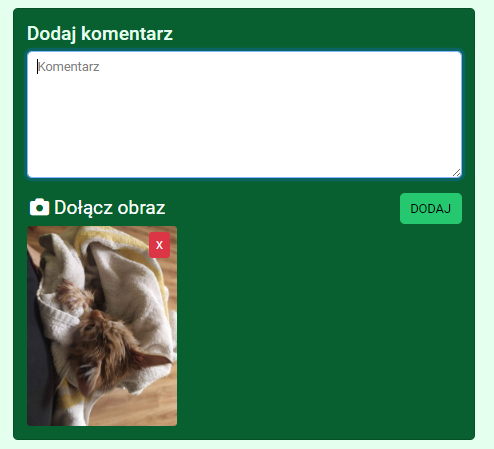
##### Wysyłanie wiadomości

Po zaakceptowaniu zaproszenia możliwe jest wysyłanie wiadomości. W liście wiadomości znajdują się wszystkie wiadomości, które są ładowane od końca mechanizmem „Infinite Scrolling”, a na końcu strony można dodać kolejną wiadomość. Ze względów na problemy techniczne z ustawianiem web socketów w Spring Boocie, nie są one tutaj używane. Zamiast tego strona co 5 sekund aktualizuje rozmowę prywatną, jeśli wystąpią jakieś zmiany.



Rysunek 0.90 Widok rozmowy prywatnej

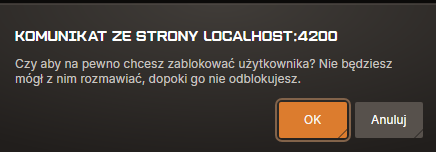
Przykładowo, poniżej zostaje wysłana wiadomość ze zdjęciem, która po dodaniu pokazuje się na liście wiadomości:

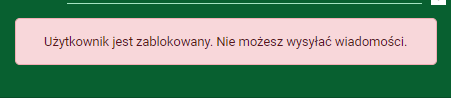


Rysunek 0.91 Dodanie wiadomości

##### Blokowanie użytkownika

Aby zablokować użytkownika, trzeba kliknąć przycisk „Zablokuj”, a następnie potwierdzić operację. Po zablokowaniu użytkownika nie jest możliwe wysyłanie do niego wiadomości, ocenianie jego postów/komentarzy ani komentowanie ich.





Rysunek 0.92 Potwierdzenie blokowania i wiadomość po zablokowaniu

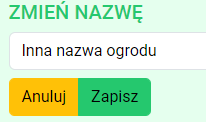
#### Zarządzanie ogrodem

Aby zarządzać ogrodem, należy w pasku nawigacyjnym wybrać opcję „Mój ogród”, gdzie w widoku ogrodu można przejść do katalogu, jednej z działek, lub zmienić nazwę ogrodu.

##### Zmiana nazwy ogrodu lub działki

Aby zmienić nazwę ogrodu lub działki, należy kliknąć ikonę po prawej od nazwy, a następnie w polu zmienić nazwę i potwierdzić:



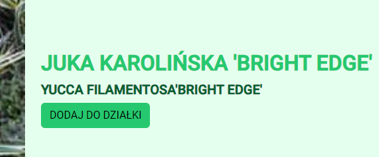




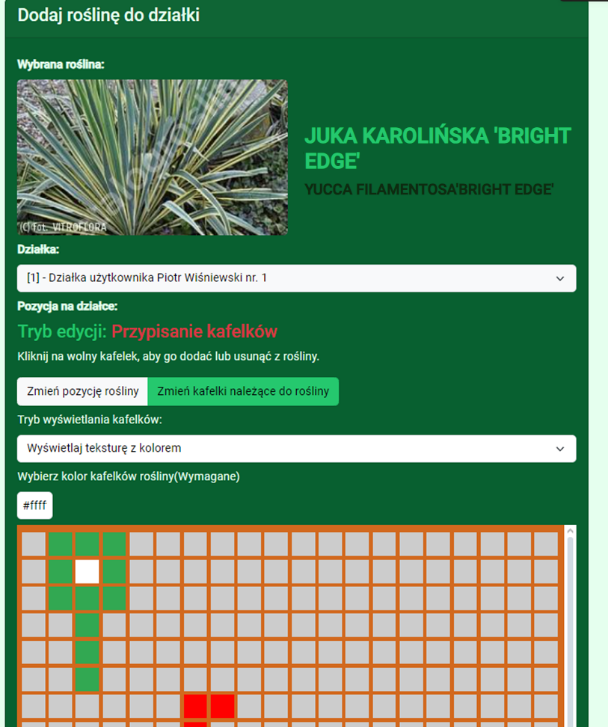
Rysunek 0.93 Zmiana nazwy ogrodu

##### Dodanie rośliny do działki

Aby dodać roślinę do działki, należy jako zalogowany użytkownik przejść do strony rośliny, a następnie nacisnąć przycisk „Dodaj do działki”, przenosząc się do strony dodawania rośliny do działki:



Rysunek 0.94 Przycisk "Dodaj do działki"



Rysunek 0.95 Strona dodawania rośliny do działki

Aby dodać roślinę do działki, należy wykonać parę kroków:

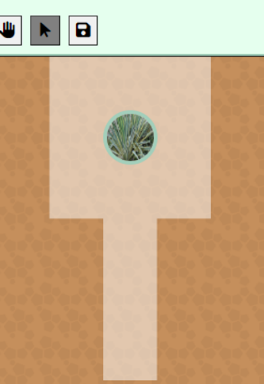
1. Wybrać działkę.
2. Wybrać tryb wyświetlania kafelków:

* Wyświetlanie samego koloru.
* Wyświetlanie samej tekstury.
* Wyświetlanie tekstury wraz z nałożonym kolorem.

1. Wybrać kafelki, które będzie zajmować roślina.

* Białe: Pozycja obrazu rośliny.
* Zielone: Zaznaczone kafelki rośliny.
* Szare: Wolne kafelki.
* Czerwone: Kafelki zajęte przez inne rośliny.

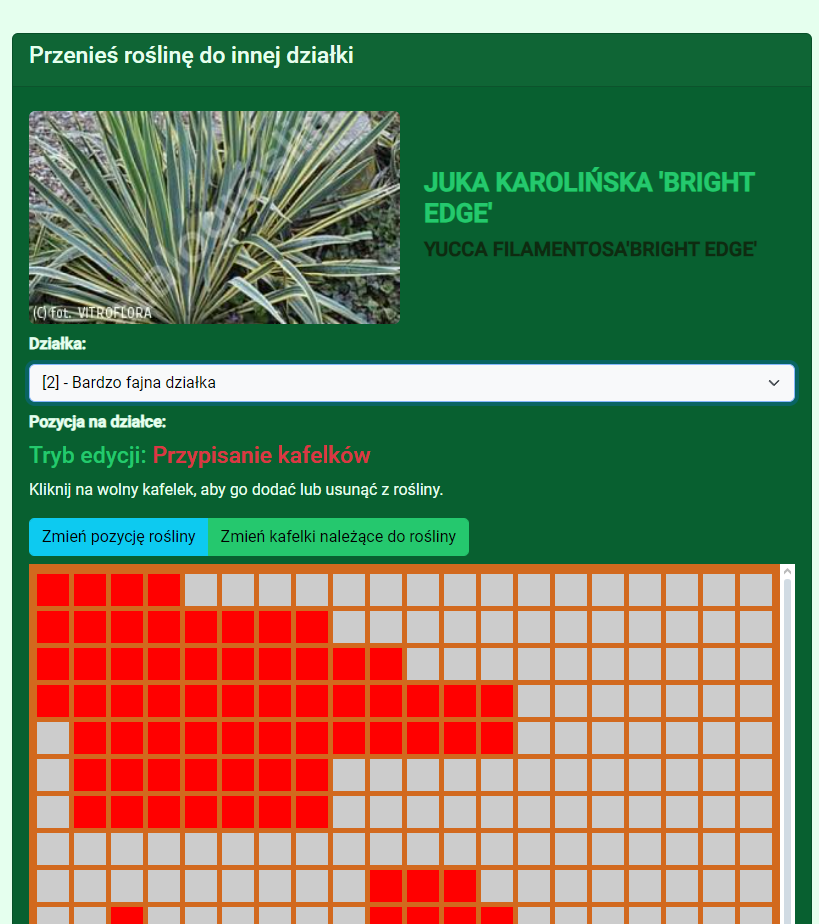
1. (Opcjonalnie) Wybrać obraz rośliny.
2. (Opcjonalnie) Wybrać teksturę kafelków. Ten krok jest wymagany, jeśli wcześniej wybrano wyświetlanie samej tekstury.
3. Wcisnąć przycisk „Dodaj roślinę”, po którym zostanie się przeniesionym do działki z nowo dodaną rośliną:



Rysunek 0.96 Widok nowo dodanej do działki rośliny

##### Przeniesienie rośliny do innej działki

Aby przenieść roślinę do innej działki, należy w trybie wybierania kliknąć na roślinę, a następnie kliknąć przycisk przenoszenia do innej działki:

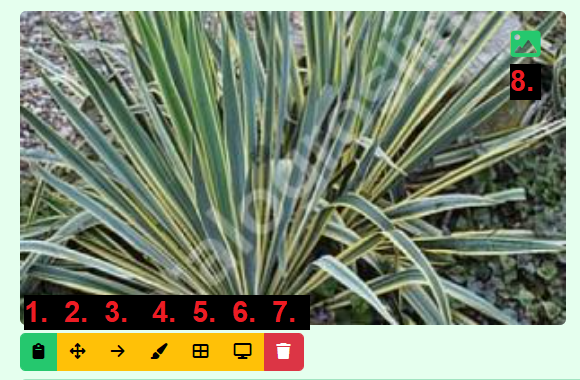


Rysunek 0.97 Przycisk przenoszenia do innej działki i widok

Następne czynności są takie same, jak w Dodawaniu rośliny do działki z tym wyjątkiem, że ustawia się jedynie pozycję rośliny na innej działce, a reszta cech pozostaje niezmieniona.

##### Zmiana właściwości rośliny

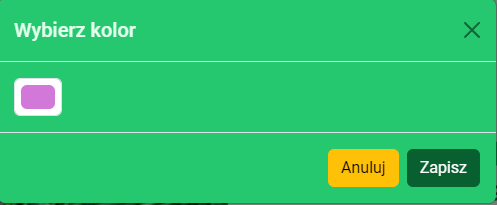
Właściwości rośliny na działce można zmieniać w jej panelu, gdzie każdy przycisk może zmienić jej właściwość:



Rysunek 0.98 Opcje zmiany właściwości rośliny

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

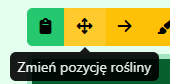
Przykładowo, można zmienić kolor kafelków rośliny, klikając na ikonę nr.4. Pojawi się wtedy okienko, gdzie wybiera się kolor, który dynamicznie zmienia się na działce po zatwierdzeniu:



Rysunek 0.99 Okno z wybieraniem koloru i roślina po zmianie

##### Zmiana pozycji rośliny na działce

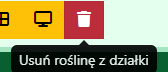
Aby zmienić pozycję rośliny na działce, należy w trybie wybierania wybrać roślinę, a następnie w jej panelu wybrać opcję „Zmień pozycję rośliny”. Tak jak w dodawaniu rośliny do działki, trzeba wybrać kafelki i pozycję rośliny, klikając na dany kafelek, a na końcu zapisać zmiany przyciskiem „Zapisz”. Po zapisaniu zmian roślina wraz z działką zostanie zaktualizowana:

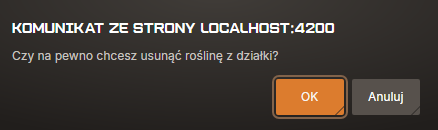
Rysunek 0.100 Zmiana pozycji rośliny

##### Usuwanie rośliny z działki

Aby usunąć roślinę z działki, należy kliknąć ikonę śmietnika i potwierdzić.



Rysunek 0.101 Usuwanie rośliny z działki



## Testy(TODO)

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

Przykładowo, KomentarzControllerTest używa rzeczywistych komponentów z kontrolera komentarzy, lecz do jego działania wymagane są istniejące posty, które są zarządzane w PostController, dlatego używa się go również w tym teście.

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

Poniżej znajduje się fragment kodu z KomentarzControllerTest, w którym jest funkcja setup, funkcja wykonująca się w danej kolejności i funkcja kończąca.

@Testcontainers

@SpringBootTest

@Slf4j

@TestMethodOrder(OrderAnnotation.class)

@TestInstance(TestInstance.Lifecycle.PER\_CLASS)

public class KomentarzControllerTest {

    @Autowired

    private PasswordEncoder passwordEncoder;

    @Autowired

    private PostController postController;

    @Autowired

    private KomentarzController komentarzController;

    @Autowired

    private UzytkownikService uzytkownikService;

    @Autowired

    private UzytkownikRepository uzytkownikRepository;

    Authentication mockAuth;

    Uzytkownik uzyt;

    Authentication otherMockAuth;

    Uzytkownik otherUzyt;

    PostResponse postResponse;

    KomentarzResponse komentarzResponse;

    @Test

    @Order(1)

    void testAddKomentarzToPost() {

        assertNotNull(postResponse.getUuid());

        KomentarzRequest request = KomentarzRequest.builder()

        .opis("Testowy opis")

        .targetId(postResponse.getUuid())

        .build();

        ResponseEntity<?> response = komentarzController.addKomentarzToPost(request, null, mockAuth);

        assertEquals(HttpStatus.CREATED, response.getStatusCode());

        KomentarzResponse body = (KomentarzResponse) response.getBody();

        assertNotNull(body);

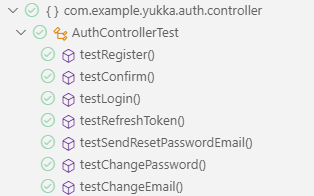
        assertEquals(request.getOpis(), body.getOpis());

        komentarzResponse = body;

    }

}

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.



Rysunek 0.103 Lista testów na backendzie

Wymień testy i daj screen z jednego testu.

# Podsumowanie(TODO)

(Wstawić podsumowanie, funkcjonalności spełnione, planowany taki rozwój i coś tam jeszcze)

# Streszczenie(TODO)

streszczenie pracy w języku polskim, tytuł pracy dyplomowej w języku polskim oraz w języku angielskim,

oświadczenie studenta o samodzielnym przygotowaniu pracy dyplomowej (załącznik do Zarządzenia w sprawie funkcjonowania procedury antyplagiatowej w Uniwersytecie Rzeszowskim).

# Spis ilustracji

[Rysunek 0.1 Węzeł Roslina i węzeł Roslina z etykietą RoslinaWlasna 10](#_Toc187618779)

[Rysunek 0.2 Węzeł Cecha i węzeł Cecha z etykietą CechaWlasna 12](#_Toc187618780)

[Rysunek 0.3 Węzeł ogrodu 12](#_Toc187618781)

[Rysunek 0.4 Węzeł Dzialka 13](#_Toc187618782)

[Rysunek 0.5 Węzeł Uzytkownik oraz węzły z etykietami Pracownik i Admin 14](#_Toc187618783)

[Rysunek 0.6 Węzeł Ustawienia 14](#_Toc187618784)

[Rysunek 0.7 Węzeł Post 15](#_Toc187618785)

[Rysunek 0.8 Węzeł Komentarz 15](#_Toc187618786)

[Rysunek 0.9 Węzeł RozmowaPrywatna 16](#_Toc187618787)

[Rysunek 0.10 Węzeł Powiadomienie bez i z etykietą Zgloszenie 17](#_Toc187618788)

[Rysunek 0.11 Relacja ZASADZONA\_NA 20](#_Toc187618789)

[Rysunek 0.12 Relacja OCENIL 21](#_Toc187618790)

[Rysunek 0.13 Relacja POWIADAMIA 21](#_Toc187618791)

[Rysunek 0.15 Struktura backendu 22](#_Toc187618792)

[Rysunek 0.16 Katalog uploads 22](#_Toc187618793)

[Rysunek 0.17 Katalog resources 23](#_Toc187618794)

[Rysunek 0.18 Żądania dla kontrolera Post w Swaggerze 24](#_Toc187618795)

[Rysunek 0.19 Widok żądania GET /posty 25](#_Toc187618796)

[Rysunek 0.20 Folder z logami 28](#_Toc187618797)

[Rysunek 0.21 Struktura frontendu 30](#_Toc187618798)

[Rysunek 0.22 Komponent roslina-card oraz jego widok 30](#_Toc187618799)

[Rysunek 0.23 Przycisk „Zaloguj się” i panel logowania 34](#_Toc187618800)

[Rysunek 0.24 Panel rejestracji 34](#_Toc187618801)

[Rysunek 0.25 Panel aktywacji i kod z wiadomości email 34](#_Toc187618802)

[Rysunek 0.28 Panel logowania z danymi i widok po zalogowaniu się 35](#_Toc187618803)

[Rysunek 0.31 Widok katalogu roślin 35](#_Toc187618804)

[Rysunek 0.33 Lista cech typu Gleba przed i po wyszukaniu 36](#_Toc187618805)

[Rysunek 0.35 Widok tagów cech 36](#_Toc187618806)

[Rysunek 0.37 Widok strony rośliny 36](#_Toc187618807)

[Rysunek 0.39 Wyszukiwarka postów i przykłady postów 37](#_Toc187618808)

[Rysunek 0.42 Widok komentarzy postu 37](#_Toc187618809)

[Rysunek 0.43 Lista użytkowników i profil jednego z nich 38](#_Toc187618810)

[Rysunek 0.50 Widok ogrodu użytkownika 38](#_Toc187618811)

[Rysunek 0.51 Widok działki 39](#_Toc187618812)

[Rysunek 0.53 Potwierdzenie i stworzony obraz działki 39](#_Toc187618813)

[Rysunek 0.45 Opcja banowania użytkownika 40](#_Toc187618814)

[Rysunek 0.46 Formularz banowania i użytkownik po zbanowaniu 40](#_Toc187618815)

[Rysunek 0.47 Komunikat o zbanowaniu przy próbie zalogowania się 40](#_Toc187618816)

[Rysunek 0.54 Przejście do ustawień i do zmiany avatara 41](#_Toc187618817)

[Rysunek 0.55 Panel zmiany avatara 41](#_Toc187618818)

[Rysunek 0.56 Profil po zmianie avatara 42](#_Toc187618819)

[Rysunek 0.57 Panel edycji profilu przed i po podaniu danych 42](#_Toc187618820)

[Rysunek 0.58 Profil po edycji 42](#_Toc187618821)

[Rysunek 0.59 Panel ustawień konta 43](#_Toc187618822)

[Rysunek 0.60 Komunikat o zmianie ustawień i profil po zmianie ustawień konta 43](#_Toc187618823)

[Rysunek 0.61 Panel zmiany adresu email 44](#_Toc187618824)

[Rysunek 0.62 Kod aktywacyjny 44](#_Toc187618825)

[Rysunek 0.63 Panel zmiany email po podaniu kodu 44](#_Toc187618826)

[Rysunek 0.64 Link do zmiany hasła i formularz 45](#_Toc187618827)

[Rysunek 0.65 Formularz zmiany hasła 45](#_Toc187618828)

[Rysunek 0.66 Udana zmiana hasła 45](#_Toc187618829)

[Rysunek 0.67 Panel usuwania konta i potwierdzenie 46](#_Toc187618830)

[Rysunek 0.68 Przycisk dodawania rośliny i formularz 46](#_Toc187618831)

[Rysunek 0.69 Nowa właściwość rośliny 47](#_Toc187618832)

[Rysunek 0.70 Dodana roślina 47](#_Toc187618833)

[Rysunek 0.71 Zmiana obrazu rośliny 48](#_Toc187618834)

[Rysunek 0.72 Potwierdzenie usuwania rośliny 48](#_Toc187618835)

[Rysunek 0.73 Przejście do postów użytkownika 48](#_Toc187618836)

[Rysunek 0.74 Formularz dodania posta i post po dodaniu 49](#_Toc187618837)

[Rysunek 0.75 Usuwanie posta 49](#_Toc187618838)

[Rysunek 0.76 Dodawanie komentarza i dodany komentarz 50](#_Toc187618839)

[Rysunek 0.77 Edycja komentarza i komentarz po edycji 50](#_Toc187618840)

[Rysunek 0.78 Dodawanie odpowiedzi i odpowiedź po edycji 50](#_Toc187618841)

[Rysunek 0.79 Ocena postu/komentarza 51](#_Toc187618842)

[Rysunek 0.80 Ikona usuwania 51](#_Toc187618843)

[Rysunek 0.81 Okno powiadomień 52](#_Toc187618844)

[Rysunek 0.82 Panel powiadomień pracownika 52](#_Toc187618845)

[Rysunek 0.83 Okno specjalnego powiadomienia i stworzone powiadomienie 53](#_Toc187618846)

[Rysunek 0.84 Przycisk zgłaszania 53](#_Toc187618847)

[Rysunek 0.85 Opis zgłoszenia i widok wysłanego zgłoszenia 54](#_Toc187618848)

[Rysunek 0.86 Przejście do rozmów 55](#_Toc187618849)

[Rysunek 0.87 Rozmowy Prywatne 55](#_Toc187618850)

[Rysunek 0.88 Przycisk "Zaproś" 55](#_Toc187618851)

[Rysunek 0.89 Przykładowe zaproszenie do rozmowy 56](#_Toc187618852)

[Rysunek 0.90 Widok rozmowy prywatnej 56](#_Toc187618853)

[Rysunek 0.91 Dodanie wiadomości 57](#_Toc187618854)

[Rysunek 0.92 Potwierdzenie blokowania i wiadomość po zablokowaniu 57](#_Toc187618855)

[Rysunek 0.93 Zmiana nazwy ogrodu 58](#_Toc187618856)

[Rysunek 0.94 Przycisk "Dodaj do działki" 58](#_Toc187618857)

[Rysunek 0.95 Strona dodawania rośliny do działki 59](#_Toc187618858)

[Rysunek 0.96 Widok nowo dodanej do działki rośliny 59](#_Toc187618859)

[Rysunek 0.97 Przycisk przenoszenia do innej działki i widok 60](#_Toc187618860)

[Rysunek 0.98 Opcje zmiany właściwości rośliny 60](#_Toc187618861)

[Rysunek 0.99 Okno z wybieraniem koloru i roślina po zmianie 61](#_Toc187618862)

[Rysunek 0.100 Zmiana pozycji rośliny 61](#_Toc187618863)

[Rysunek 0.101 Usuwanie rośliny z działki 61](#_Toc187618864)

[Rysunek 0.103 Lista testów na backendzie 63](#_Toc187618865)