

AI1	Dokumentacja projektu
Autor	Krzysztof Motas, 125145
Kierunek, rok	Informatyka, II rok, st. stacjonarne (3,5-l)
Temat projektu	<i>OnlyPets – portal społecznościowy dla zwierząt</i>

1. Wstęp

Tematem mojego projektu jest stworzenie aplikacji o nazwie OnlyPets, która odwzorowuje funkcjonalności znanej platformy OnlyFans, ale jest przeznaczona do publikowania zdjęć zwierząt pod subskrypcją. Głównym założeniem aplikacji jest umożliwienie użytkownikom tworzenia i przeglądania treści dotyczących zwierząt w interaktywny i prosty sposób w obsłudze.

1.1. Zakres i funkcje aplikacji

Aplikacja obejmuje fragment rzeczywistości związany z miłośnikami zwierząt, którzy chcą dzielić się zdjęciami swoich pupili z innymi użytkownikami. Aplikacja umożliwia:

- **Logowanie i rejestrację użytkowników:** użytkownicy mogą w każdej chwili założyć konto lub zalogować się do istniejącego konta.
- **Publikowanie postów:** zarejestrowani i zalogowani użytkownicy mogą tworzyć posty, które mogą zawierać tekst oraz zdjęcia ich zwierząt.
- **Subskrypcje:** użytkownicy mogą uzyskać dostęp do treści innych użytkowników poprzez wykupienie czasowej subskrypcji. Subskrypcja jest jednorazowa i może być przedłużona przed jej wygaśnięciem.
- **Przeglądanie treści:** posty są wyświetlane na stronie głównej od kolejności od najnowszych, a użytkownik może eksplorować zasoby strony poprzez scrollowanie. Aplikacja implementuje technologię endless scrolling, znaną z X (Twitter), co umożliwia ładowanie kolejnych postów podczas przewijania strony.
- **Zarządzanie kontem:** użytkownicy mają możliwość personalizacji swoich profili, w tym dodawania opisu, linku do strony internetowej, miejscowości, ustawienia awatara oraz tła profilowego. W ustawieniach dostępne są także opcje zmiany hasła, nazwy użytkownika i innych danych.
- **Interakcje z treścią:** na profilach użytkowników wyświetlane są ich posty oraz różne elementy informacyjne, takie jak statystyki. Użytkownicy bez subskrypcji są poinformowani o konieczności jej wykupienia, aby uzyskać dostęp do pełnej zawartości.
- **Wyszukiwanie profili użytkowników:** dla ułatwienia korzystania z portalu, użytkownicy mają możliwość wyszukiwania innych użytkowników.

2. Narzędzia i technologie

Projekt został zrealizowany przy użyciu narzędzi i technologii. Wykorzystane technologie obejmują różne języki programowania, bazę danych, biblioteki oraz framework.

Poniżej przedstawiono opis tych technologii, w tym linki do ich dokumentacji oraz informacje na temat licencji i dostępności.

2.1. Języki programowania

- **PHP:**
 - Wersja: PHP 8.2.12
 - Link do dokumentacji: <https://www.php.net/docs.php>
 - Licencja: <https://www.php.net/license/index.php>
 - Dostępność: darmowy
- **JavaScript:**
 - Wersja: ECMAScript 2023
 - Link do dokumentacji: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
 - Dostępność: darmowy

2.2. Języki znaczników

- **HTML:**
 - Wersja: HTML 5
 - Link do dokumentacji: <https://developer.mozilla.org/en-US/docs/Web/HTML>
 - Dostępność: darmowy

2.3. Języki arkuszy stylów

- **CSS:**
 - Wersja: CSS 5
 - Link do dokumentacji: <https://developer.mozilla.org/en-US/docs/Web/CSS>
 - Dostępność: darmowy

2.4. Narzędzia

- **XAMPP:**
 - Wersja: 8.2.12
 - Link: <https://www.apachefriends.org/pl/index.html>
 - Licencja: <https://www.apachefriends.org/pl/about.html>
 - Dostępność: darmowy
- **Composer:**
 - Wersja: 2.7.6
 - Link do dokumentacji: <https://getcomposer.org/doc/>
 - Licencja: <https://github.com/composer/composer/blob/main/LICENSE>
 - Dostępność: darmowy

2.5. Biblioteki i frameworki

- **Laravel:**
 - Wersja: 11.0.9
 - Link do dokumentacji: <https://laravel.com/docs/11.x/readme>
 - Licencja: <https://github.com/translation/laravel/blob/master/LICENSE>
 - Dostępność: darmowy
- **Bootstrap:**
 - Wersja: 5.3.3
 - Link do dokumentacji: <https://getbootstrap.com/>
 - Licencja: <https://getbootstrap.com/docs/5.3/about/license/>
 - Dostępność: darmowa
- **Emoji Mart:**
 - Wersja: 5.6.0
 - Link do dokumentacji: <https://github.com/missive/emoji-mart>
 - Licencja: <https://github.com/missive/emoji-mart?tab=MIT-1-ov-file#readme>
 - Dostępność: darmowa

2.6. Bazy danych

- **MySQL:**
 - Wersja: 11.0.9
 - Link do dokumentacji: <https://dev.mysql.com/doc/>
 - Licencja: <https://www.mysql.com/about/legal/licensing/oem/>
 - Dostępność: darmowy

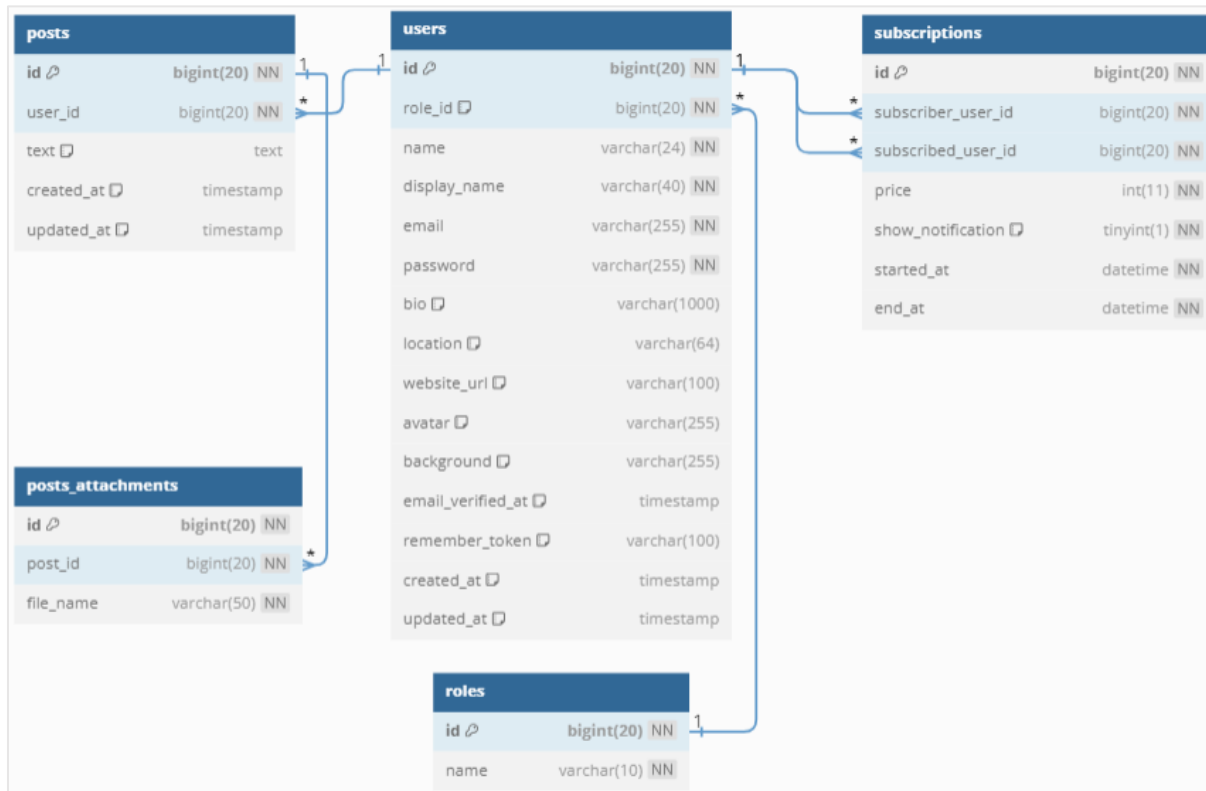
2.7. Środowiska programistyczne

- **Visual Studio Code:**

- Wersja: 1.89.1
- Link: <https://code.visualstudio.com/>
- Licencja: <https://code.visualstudio.com/License/>
- Dostępność: darmowy

3. Baza danych

Poniżej znajduje się schemat ERD dla bazy danych aplikacji.



Zrzut ekranu 1 - Diagram ERD bazy danych

3.1. Opis zawartości bazy danych

- **Tabela "posts"**: Przechowuje informacje o postach użytkowników.
 - Kolumny: id, user_id, text, created_at, updated_at
 - Powiązanie: user_id jest kluczem obcym, który odnosi się do id w tabeli "users".
- **Tabela "posts_attachments"**: Przechowuje informacje o załącznikach do postów.
 - Kolumny: id, post_id, file_name
 - Powiązanie: post_id jest kluczem obcym, który odnosi się do id w tabeli "posts".
- **Tabela "roles"**: Przechowuje informacje o rolach użytkowników.
 - Kolumny: id, name
- **Tabela "subscriptions"**: Przechowuje informacje o subskrypcjach między użytkownikami.
 - Kolumny: id, subscriber_user_id, subscribed_user_id, price, show_notification, started_at, end_at

- Powiązania: subscriber_user_id jest kluczem obcym, który odnosi się do id w tabeli "users", subscribed_user_id jest kluczem obcym, który odnosi się do id w tabeli "users".
- **Tabela "users":** Przechowuje informacje o użytkownikach.
- Kolumny: id, role_id, name, display_name, email, password, bio, location, website_url, avatar, background, email_verified_at, remember_token, created_at, updated_at
- Powiązanie: role_id jest kluczem obcym, który odnosi się do id w tabeli "roles".

4. GUI

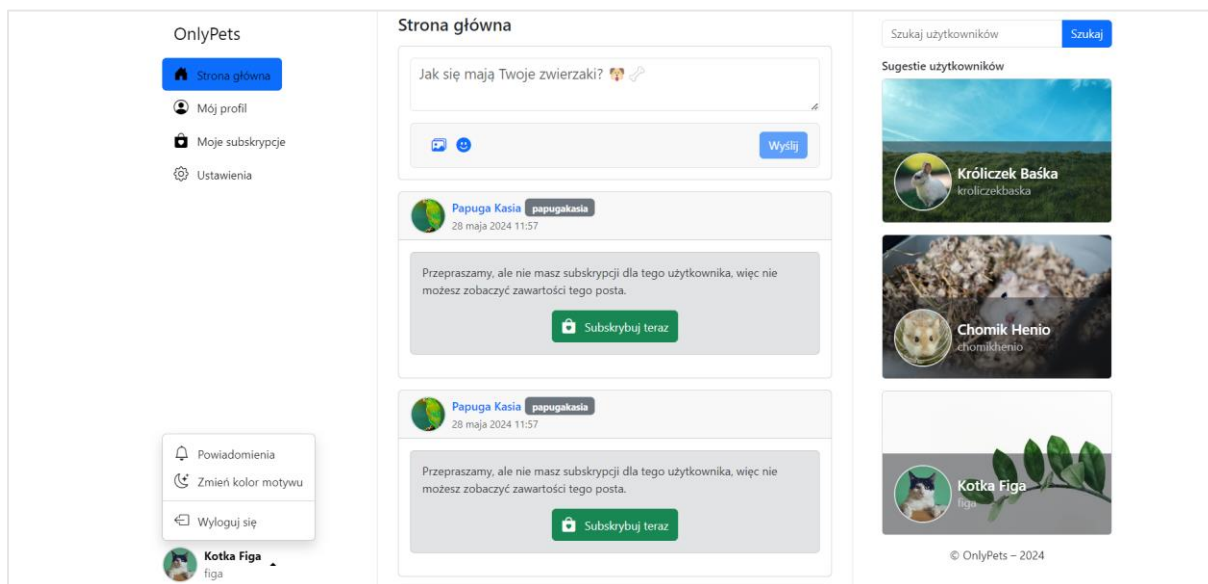
Poniżej są zaprezentowane trzy kluczowe widoki aplikacji.

- **Widok strony głównej:**

Strona główna aplikacji jest podzielona na trzy kolumny. W pierwszej kolumnie znajduje się nawigacja, która ułatwia poruszanie się po stronie internetowej. Zawiera ona również opcje użytkownika, takie jak powiadomienia, zmiana koloru motywu oraz wylogowanie. Pierwsza kolumna jest dostępna na każdej podstronie aplikacji.

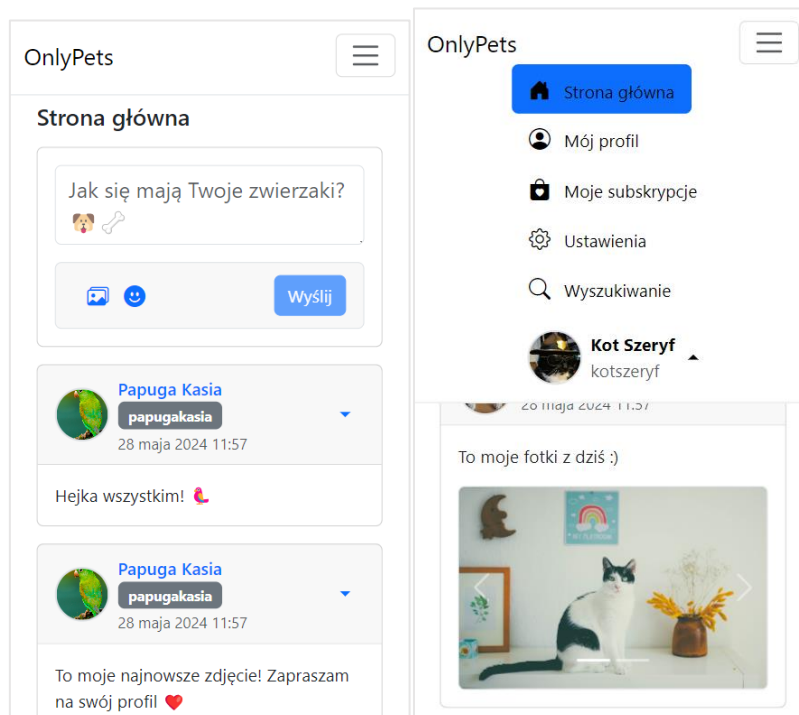
W środkowej kolumnie znajduje się formularz tworzenia nowego posta, który oferuje opcje edycji tekstu, dodawania emoji oraz przycisk "Wyślij". Poniżej formularza wyświetlane są posty użytkowników, zawierające informacje takie jak nazwa użytkownika, nazwa wyświetlana, awatar, treść posta, załączniki, data utworzenia i edycji posta. Jeśli użytkownik nie posiada subskrypcji dla autora danego posta, wyświetlana jest informacja o konieczności subskrypcji.

W trzeciej kolumnie umieszczony jest formularz wyszukiwania użytkowników, a poniżej znajdują się sugestie użytkowników, które służą do promocji członków aplikacji.



Zrzut ekranu 2 - Widok strony głównej

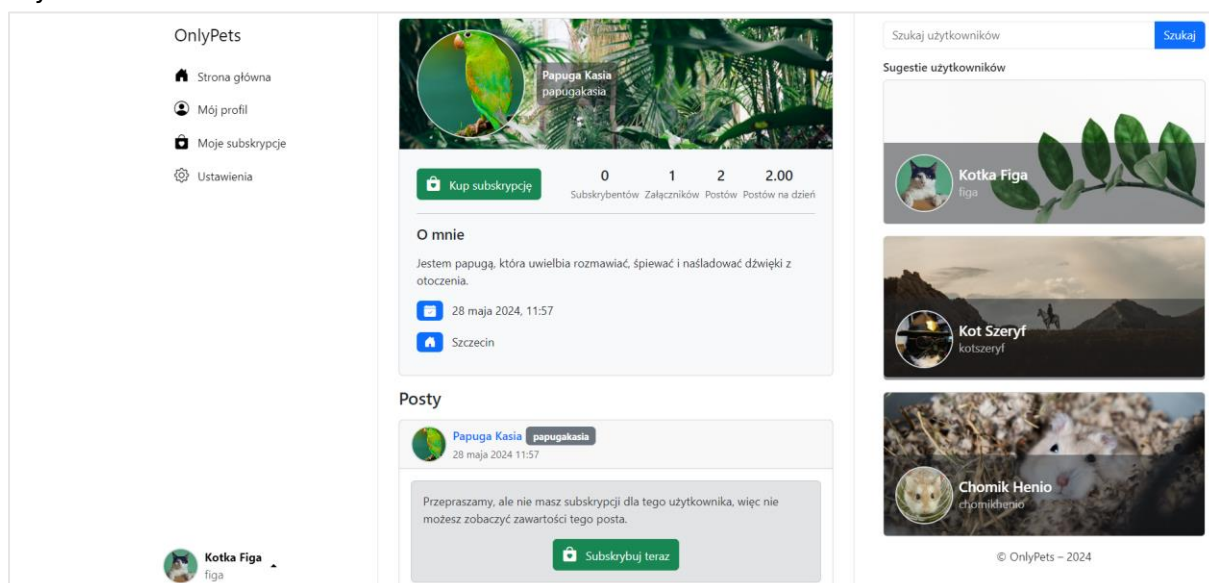
Strona główna aplikacji jest przystosowana do przeglądania na urządzeniach mobilnych. Nawigacja jest zawsze dostępna, ponieważ jest przyklejona do góry ekranu i można ją rozwinąć, naciskając przycisk. Wszystkie elementy są widoczne i funkcjonalne, zapewniając użytkownikom wygodne korzystanie z aplikacji na różnych urządzeniach.



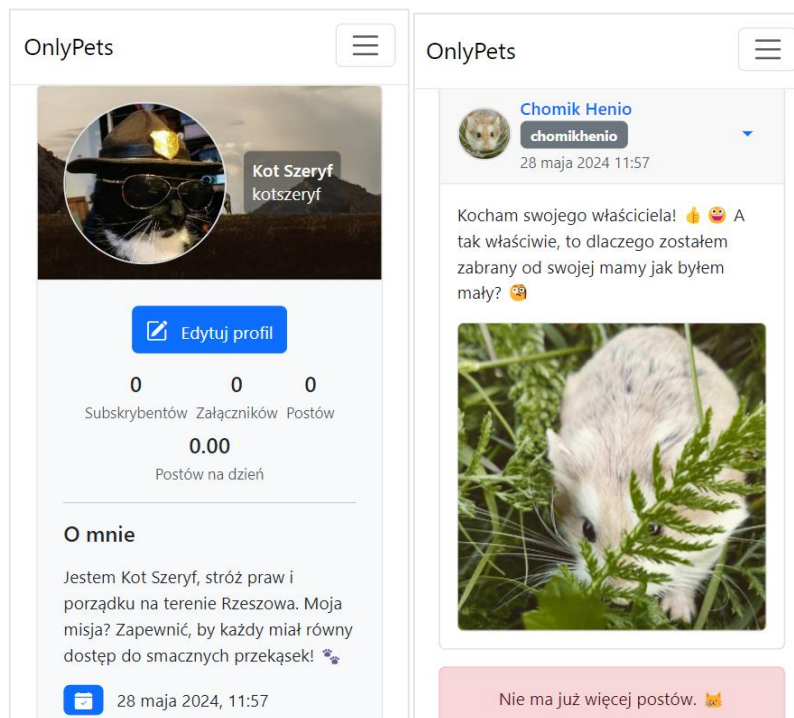
Zrzut ekranu 3 – Widok strony głównej w wersji mobilnej

- **Widok profilu użytkownika:**

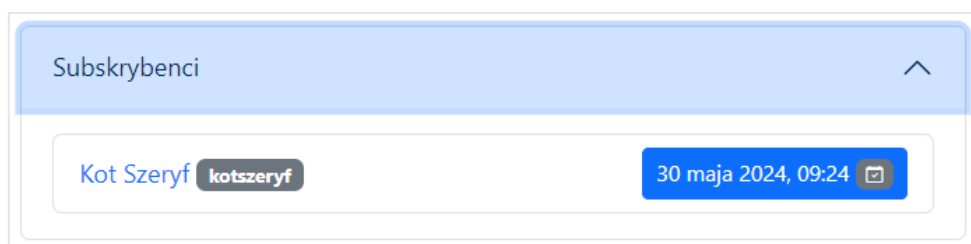
W widoku profilu użytkownika, w środkowej kolumnie znajdują się takie informacje jak nazwa użytkownika, jego awatar, zdjęcie w tle oraz statystyki (liczba subskrybentów, załączników, postów, postów na dzień). Widoczny jest również personalizowany opis użytkownika, data rejestracji i miejscowość. Jeśli użytkownik przegląda swój własny profil, może zobaczyć w tym miejscu listę swoich subskrybentów. Poniżej sekcji informacyjnej wyświetlane są posty użytkownika.



Zrzut ekranu 4 - Widok profilu użytkownika



Zrzut ekranu 5 - Widok profilu użytkownika w wersji mobilnej



Zrzut ekranu 6 - Rozwijalna zakładka przedstawiająca subskrybentów

- **Widok strony kupna subskrypcji:**

Na stronie zakupu nowej subskrypcji znajdują się informacje o korzyściach, jakie użytkownik uzyska po jej zakupie. Użytkownik jest również informowany o podziale płatności, w tym o tym, jaka część jego wpłaty trafi do twórcy. Strona zawiera także ustawienia subskrypcji, takie jak długość jej ważności. W dolnej części formularza znajdują się pola wymagane do dokonania płatności, takie jak imię, nazwisko, numer karty kredytowej i inne niezbędne dane.

The screenshot shows a desktop web interface for 'OnlyPets'. On the left is a sidebar with navigation links: 'Strona główna', 'Mój profil', 'Moje subskrypcje' (highlighted in blue), and 'Ustawienia'. Below the links is a user profile for 'Kotka Figa'. The main content area is titled 'Kupno nowej subskrypcji'. It includes an 'Informacja' section with details about the subscription benefits and payment split. Below this is the 'Ustawienia subskrypcji' section with a dropdown for 'Długość subskrypcji' set to '1 miesiąc' and a text box for 'Cena subskrypcji' showing '10 zł'. The 'Płatność' section contains fields for 'Imię i nazwisko na karcie' (Kotka Figa), 'Numer karty kredytowej' (4242424242424), 'Data ważności' (12/34), and 'CVV' (568). A blue button 'Przejdź do płatności' is at the bottom.

Zrzut ekranu 7 - Widok strony kupna subskrypcji

The screenshot shows the mobile version of the 'OnlyPets' subscription purchase page. It features a hamburger menu icon in the top left. The layout is vertical. The 'Informacja' section is at the top, followed by the 'Ustawienia subskrypcji' section with a dropdown for 'Długość subskrypcji' set to '1 miesiąc' and a text box for 'Cena subskrypcji' showing '10 zł'. Below this is the 'Płatność' section with fields for 'Imię i nazwisko na karcie' (Kot Szeryf), 'Numer karty kredytowej' (4242424242424), 'Data ważności' (12/34), and 'CVV' (567). A blue button 'Przejdź do płatności' is at the bottom.

Zrzut ekranu 8 - Widok strony kupna subskrypcji w wersji mobilnej

- **Widok ustawień użytkownika:**

W sekcji ustawień użytkownik może znaleźć opcje zmiany różnych danych, takich jak nazwa, nazwa wyświetlana, hasło, awatar, zdjęcie w tle oraz inne informacje.

The screenshot shows a desktop web interface for 'OnlyPets'. On the left is a sidebar with navigation links: 'Strona główna', 'Mój profil', 'Moje subskrypcje', and 'Ustawienia' (highlighted with a blue button). The main content area is titled 'Ustawienia' and contains a horizontal menu with icons and labels for 'Nazwa', 'Nazwa wyświetlana', 'Hasło', 'Awatar', 'Zdjęcie w tle', and 'Inne informacje' (the last one is highlighted with a blue button). Below this menu, the 'Inne informacje' section is active, showing a bio text area with the text 'Jestem ciekawską kotką, która uwielbia wspinaczki, polowania na zabawki i długie drzemki na kanapie.', a location field with 'Łódź', and an internet address field. A blue 'Zmień swoje dane' button is at the bottom. At the bottom left of the page, there is a user profile card for 'Kotka Figa' with a small avatar.

Zrzut ekranu 9 - Widok ustawień użytkownika

The image shows two side-by-side screenshots of the mobile app interface for 'OnlyPets'. Both screens have a hamburger menu icon in the top right corner. The left screen shows the 'Ustawienia' page with a list of settings: 'Nazwa' (highlighted with a blue button), 'Nazwa wyświetlana', 'Hasło', 'Awatar', 'Zdjęcie w tle', and 'Inne informacje'. Below the list, the 'Nazwa' field contains the text 'kotszeryf' and a blue 'Zmień nazwę' button. The right screen shows the 'Ustawienia' page with the same list, but 'Inne informacje' is highlighted with a blue button. Below it, the bio text area contains 'Jestem Kot Szeryf, stróż praw i porządku na terenie Rzeszowa. Moja...' and is highlighted with a blue border. Below the bio are fields for 'Miejscowość' (containing 'Rzeszów') and 'Adres strony internetowej'. A blue 'Zmień swoje dane' button is at the bottom.

Zrzut ekranu 10 - Widok ustawień użytkownika w wersji mobilnej

5. Uruchomienie aplikacji

Przed uruchomieniem aplikacji wymagane jest wykonanie kroków, zawartych poniżej.

5.1. XAMPP: instalacja XAMPP 8.2.12, a następnie uruchomienie XAMPP Control Panel, włączenie Apache oraz MySQL. Zalogowanie się do phpMyAdmin, a następnie utworzenie bazy danych „onlypets”.

5.2. Composer:

instalacja Composer 2.7.6 według instrukcji <https://getcomposer.org/doc/00-intro.md>

5.3. Polecenia: otworenie terminala, następnie przejście do katalogu aplikacji Laravel i wykonanie następujących komend:

- **Instalacja wszystkich zależności aplikacji:** composer install
- **Wykonanie wszystkich migracji bazy danych, a także wypełnienie bazy danych przykładowymi danymi:** php artisan migrate:fresh --seed
- **Wygenerowanie unikalnego klucza aplikacji:** php artisan key:generate
- **Stworzenie symbolicznego link między katalogiem i przechowywania, a katalogiem publicznym:** php artisan storage:link.
- **Uruchomienie wbudowanego serwera deweloperskiego:** php artisan serve

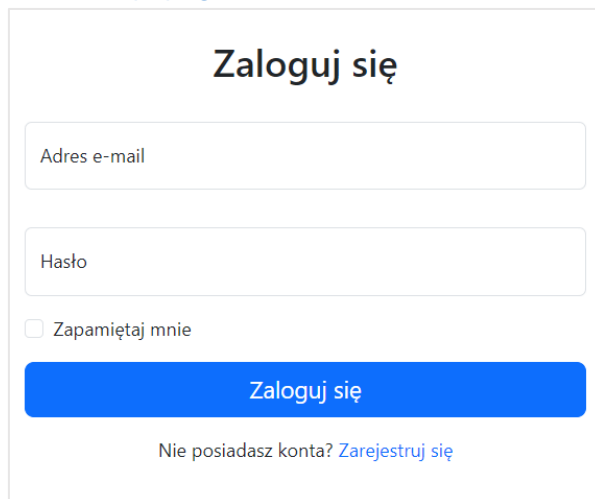
5.4. Przejście na stronę: otworenie przeglądarki internetowej i przejście na pod adres `http://localhost:8000.` (lub inny adres, jeśli port jest inny).

6. Funkcjonalności aplikacji

6.1. Logowanie i rejestracja

Do zalogowania do aplikacji wymagane jest podanie adresu e-mail oraz hasła. Przykładowe konta, na które można się zalogować:

- **kotszeryf** (konto administratora): adres e-mail: kotszeryf@email.com, hasło: 1234
- **chomikhenio**: adres e-mail: chomikhenio@email.com, hasło: 1234
- **papugakasia**: adres e-mail: papugakasia@email.com, hasło: 1234



Zrzut ekranu 11 - Okno logowania

```
public function login()
{
    return view('auth.login');
}

public function handle(Request $request)
{
    $credentials = $request->validate([
        'email' => ['required', 'email', 'max:255'],
        'password' => ['required', 'min:4'],
    ]);

    $remember = $request->has('remember');

    if (Auth::attempt($credentials, $remember)) {
        $request->session()->regenerate();

        return redirect()
            ->route('home')
            ->with('successToast', 'Zostałeś pomyślnie zalogowany!');
    }

    return back()
        ->withErrors(['email' => 'Nieprawidłowe dane logowania. Sprawdź wprowadzone informacje i spróbuj ponownie.'])
        ->onlyInput('email');
}
```

Zrzut ekranu 12 - Implementacja walidacji i procesu logowania

```
<input id="email" type="email" class="form-control @error('email') is-invalid @enderror"
    name="email" value="{{ old('email') }}" required autocomplete="email" placeholder="Adres e-mail"
    maxlength="255">
```

Zrzut ekranu 13 - Walidacja adresu e-mail po stronie Front-end

```
<input id="password" type="password" class="form-control @error('password') is-invalid @enderror"
    name="password" required autocomplete="current-password" placeholder="Hasło" minlength="4">
```

Zrzut ekranu 14 - Walidacja hasła po stronie Front-end

Zaloguj się

Adres e-mail
piesmiska\$email.com

! Uwzględnij znak „@” w adresie e-mail. W adresie „piesmiska\$email.com” brakuje znaku „@”.

Hasło
..

☒ Zapamiętaj mnie

Zaloguj się

Nie posiadasz konta? [Zarejestruj się](#)

Zrzut ekranu 15 - Działanie walidacji po stronie Front-endu dla adresu e-mail

Zaloguj się

Adres e-mail
piesmiska@email.com

Hasło
..

! Wydluż ten tekst przynajmniej do 4 znaków (teraz używasz 2 znaków).

Zaloguj się

Nie posiadasz konta? [Zarejestruj się](#)

Zrzut ekranu 16 - Działanie walidacji po stronie Front-endu dla hasła

Zaloguj się

Adres e-mail
piesmiska@email.com

Hasło

Pole hasło musi mieć przynajmniej 4 znaków.

☒ Zapamiętaj mnie

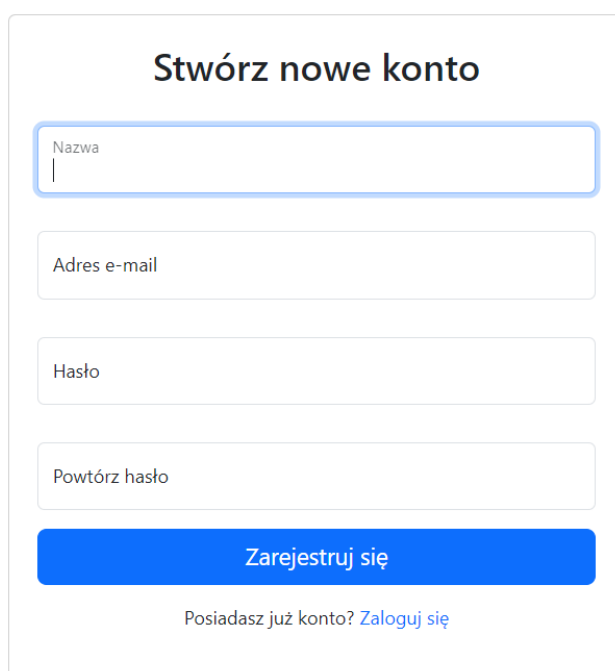
Zaloguj się

Nie posiadasz konta? [Zarejestruj się](#)

Zrzut ekranu 17 - Działanie walidacji po stronie Back-endu dla hasła

Walidacja po stronie front-endu odbywa się za pomocą atrybutów HTML, takich jak "required", "minlength", "maxlength", które są umieszczone w polach formularza.

Na przykład, atrybut "required" sprawia, że pole jest wymagane do wypełnienia przed wystaniem formularza, natomiast "minlength" i "maxlength" określają minimalną i maksymalną długość tekstu, jaki użytkownik może wprowadzić w polu. Natomiast walidacja po stronie back-endu odbywa się w kodzie serwera, gdzie dane z formularza są sprawdzane pod kątem zgodności z określonymi regułami. Na przykład, w kodzie PHP, wykorzystując metodę "validate" obiektu "Request", definiuje się reguły walidacji dla poszczególnych pól formularza. Te reguły, takie jak "required", "email", "min:4", są używane do określenia wymagań dotyczących danych wejściowych. Jeśli dane nie spełniają określonych reguł, generowany jest błąd, który jest obsługiwany w widoku, informując użytkownika o błędnych danych.



Stwórz nowe konto

Nazwa

Adres e-mail

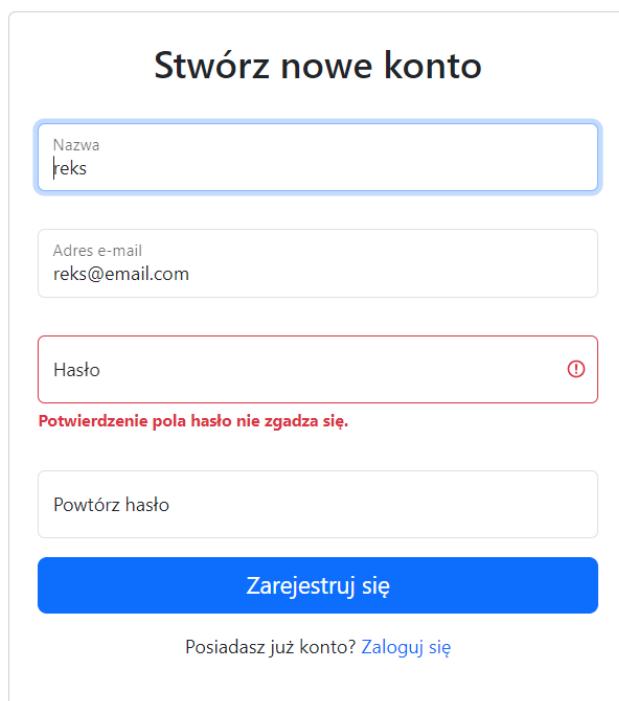
Hasło

Powtórz hasło

Zarejestruj się

Posiadasz już konto? [Zaloguj się](#)

Zrzut ekranu 18 - Okno rejestracji



Stwórz nowe konto

Nazwa
reks

Adres e-mail
reks@email.com

Hasło

Potwierdzenie pola hasło nie zgadza się.

Powtórz hasło

Zarejestruj się

Posiadasz już konto? [Zaloguj się](#)

Zrzut ekranu 19 - Działanie walidacji potwierdzenia hasła po stronie Back-endu

```

public function handle(Request $request)
{
    $validatedData = $request->validate([
        'display_name' => ['required', 'string', 'max:40'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string', 'min:4', 'confirmed'],
    ]);

    $displayName = $validatedData['display_name'];

    $user = User::create([
        'name' => User::generateUniqueName($displayName),
        'display_name' => $displayName,
        'email' => $validatedData['email'],
        'password' => Hash::make($validatedData['password'])
    ]);

    Auth::login($user);
    $request->session()->regenerate();

    return redirect()
        ->route('home')
        ->with('successToast', 'Zostałeś pomyślnie zarejestrowany!');
}

```

Zrzut ekranu 20 - Implementacja obsługi i walidacji okna rejestracji po stronie Back-endu

```

<input id="display_name" type="text" class="form-control @error('display_name') is-invalid @enderror"
    name="display_name" value="{{ old('display_name') }}" required autocomplete="name" autofocus
    placeholder="Nazwa" maxlength="40">

```

Zrzut ekranu 21 - Walidacja nazwy użytkownika po stronie Front-endu

```

<input id="email" type="email" class="form-control @error('email') is-invalid @enderror"
    name="email" value="{{ old('email') }}" required autocomplete="email" placeholder="Adres e-mail"
    maxlength="255">

```


Zrzut ekranu 22 - Walidacja adresu e-mail użytkownika po stronie Front-endu


Walidacja po stronie Front-endu, jaki Back-endu odbywa się na analogicznej zasadzie, jak w przypadku logowania.


6.2. Zarządzanie swoimi danymi przez użytkownika aplikacji


Po kliknięciu przycisku "Ustawienia", użytkownik ma możliwość zarządzania swoimi danymi w aplikacji. W tej może zmienić swoją nazwę, nazwę wyświetlaną, awatar, zdjęcie w tle, hasło oraz inne informacje osobiste.


Ustawienia


 Nazwa

 Nazwa wyświetlana

 Hasło

 Awatar

 Zdjęcie w tle

 Inne informacje

Nazwa

sarapies

Zmień nazwę

Zrzut ekranu 23 - Formularz zmiany nazwy użytkownika

Ustawienia

 Nazwa  **Nazwa wyświetlana**  Hasło  Awatar  Zdjęcie w tle  Inne informacje

Nazwa wyświetlana
sarapies

Zmień nazwę wyświetlaną

Zrzut ekranu 24 - Formularz zmiany nazwy wyświetlanej użytkownika

Ustawienia

 Nazwa  Nazwa wyświetlana  **Hasło**  Awatar  Zdjęcie w tle  Inne informacje

Aktualne hasło

Nowe hasło

Powtórz nowe hasło

Zmień hasło

Zrzut ekranu 25 - Formularz zmiany hasła użytkownika

Ustawienia

 Nazwa  Nazwa wyświetlana  Hasło  **Awatar**  Zdjęcie w tle  Inne informacje

Wybierz plik Nie wybrano pliku

Usuń aktualny awatar



Zrzut ekranu 26 - Formularz zmiany awatara użytkownika

Ustawienia

 Nazwa  Nazwa wyświetlana  Hasło  Awatar  Zdjęcie w tle  Inne informacje

Wybierz plik

Nie wybrano pliku

Usuń aktualne zdjęcie w tle



Zrzut ekranu 27 - Formularz zmiany zdjęcia w tle użytkownika

Ustawienia

 Nazwa  Nazwa wyświetlana  Hasło  Awatar  Zdjęcie w tle  Inne informacje

Bio
Hejka naklejka, zapraszam do
subowania! 😊

Miejscowość
Berlin

Adres strony internetowej

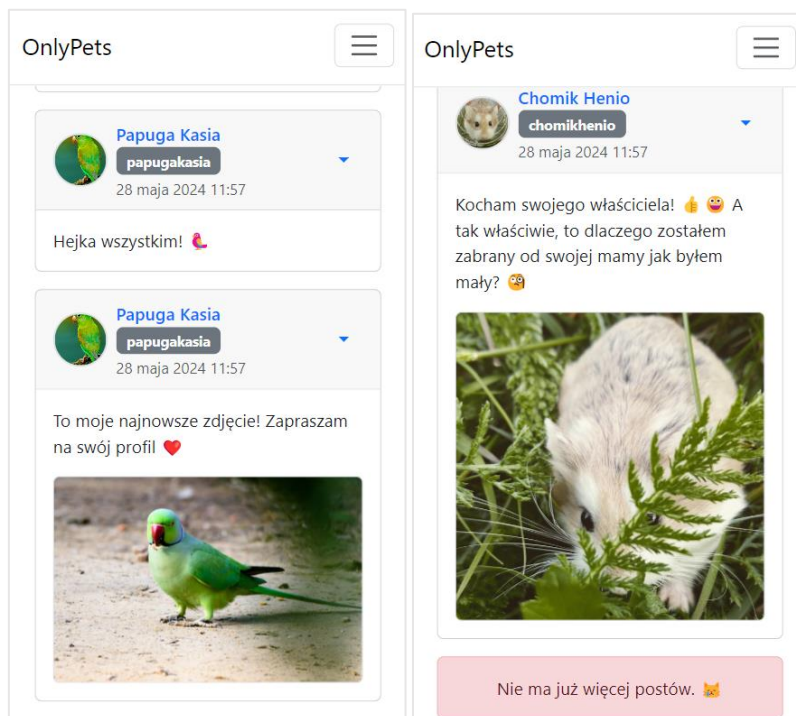
Zmień swoje dane

Zrzut ekranu 28 - Formularz zmiany pozostałych informacji użytkownika

6.3. Przeglądanie ogólnodostępnych zasobów

Użytkownicy mają możliwość przeglądania zasobów oraz ich obrazków dostępnych w aplikacji. Zasoby te są wyświetlane na stronie głównej. Dostęp do zawartości konkretnego postu zależy od posiadania subskrypcji na autora tego postu. W przypadku braku subskrypcji, użytkownikowi zostanie wyświetlona informacja o konieczności jej zakupu. Administrator strony ma dostęp do wszystkich zasobów bez konieczności posiadania subskrypcji.

W wyświetlaniu postów zastosowano technikę endless scrolling, znaną z Twittera. Mechanizm ten polega na pobieraniu zawartości z bazy danych w miarę przewijania strony przez użytkownika. Dzięki wykorzystaniu funkcjonalności paginacji Laravela oraz kodu JavaScript, strona w zoptymalizowany sposób prezentuje treści użytkownikom.



Zrzut ekranu 29 - Widok zasobów użytkowników

```
public function index(Request $request)
{
    if ($request->ajax()) {
        $posts = Post::with([
            'user:id,name,display_name,avatar',
            'attachments:post_id,file_name'
        ])->orderBy('created_at', 'desc')->paginate(10);

        /** @var \App\Models\User $user */
        $user = Auth::user();
        $posts = self::processPosts($posts, $user);

        return response()->json([
            'posts' => $posts,
            'next_page_url' => $posts->nextPageUrl()
        ]);
    }

    $suggestedUsers = User::getSuggestedUsers();
    return view('home.index', compact('suggestedUsers'));
}
```

Zrzut ekranu 30 - Implementacja metody index

```

private static function processPosts($posts, $user = null)
{
    $posts->transform(function ($post) use ($user) {
        if ($user) {
            $post->is_subscribed = ($user->id == $post->user->id || $user->isAdmin())
            ? true : $user->hasActiveSubscriptionFor($post->user_id);
        } else {
            $post->is_subscribed = false;
        }

        if (!$post->is_subscribed) {
            unset($post->text);
            unset($post->attachments);
        }

        return $post;
    });

    return $posts;
}

```

Zrzut ekranu 31 - Implementacja metody statycznej processPosts

```

window.addEventListener('scroll', () => {
    // console.log(`${window.innerHeight} + ${window.pageYOffset} >= ${document.body.scrollHeight}`);

    if (!noMorePosts && !loadingPosts &&
        (window.innerHeight + window.pageYOffset) >= document.body.scrollHeight - 100) {
        loadingPosts = true;

        page++;
        loadMorePosts();
    }
});

```

Zrzut ekranu 32 - Implementacja następowania zdarzenia scrollingu w JavaScript

```

function loadMorePosts() {
    showLoadingSpinner();

    const url = `?page=${page}`;
    // console.log(`loadMorePosts -> page: ${page}`);

    fetch(url, {
        headers: {
            'X-Requested-With': 'XMLHttpRequest',
        }
    })
    .then(response => response.json())
    .then(data => {
        spinner.remove();

        if (data.posts.data.length === 0) {
            const warningTemplate = document.getElementById('post-warning-template');
            const warning = document.importNode(warningTemplate.content, true);

            postsContainer.appendChild(warning);

            noMorePosts = true;
            return;
        }

        const isAdmin = @json(optional(Auth::user())->isAdmin());

        for (const post of data.posts.data) {
            const newPost = document.importNode(postTemplate.content, true);

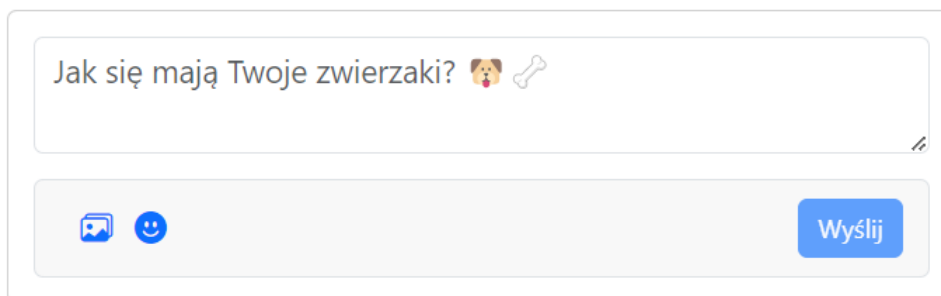
```

Zrzut ekranu 33 - Implementacja wysyłania ządania i odbierania danych zasobów w JavaScript

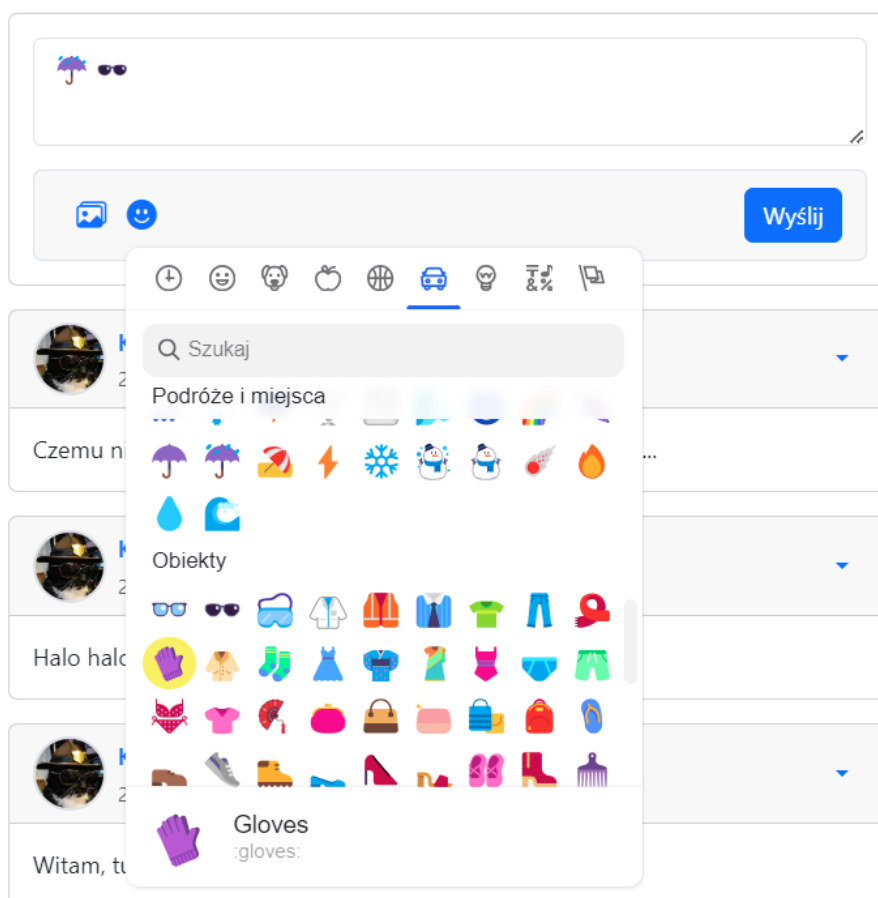
Kod funkcji PHP obsługuje zarówno ządania AJAX, jak i standardowe ządania HTTP w aplikacji Laravel. Jeśli ządanie jest AJAX, pobiera posty wraz z informacjami o użytkownikach i załącznikach, a następnie przetwarza je, aby sprawdzić, czy użytkownik ma aktywne subskrypcje. Na końcu zwraca dane w formacie JSON. W przypadku standardowego ządania HTTP, renderuje stronę główną.

W części JavaScriptowej, dodano obsługę przewijania strony. Gdy użytkownik przewinie stronę blisko końca, następuje automatyczne ładowanie kolejnych postów poprzez AJAX.

6.4. Tworzenie postów

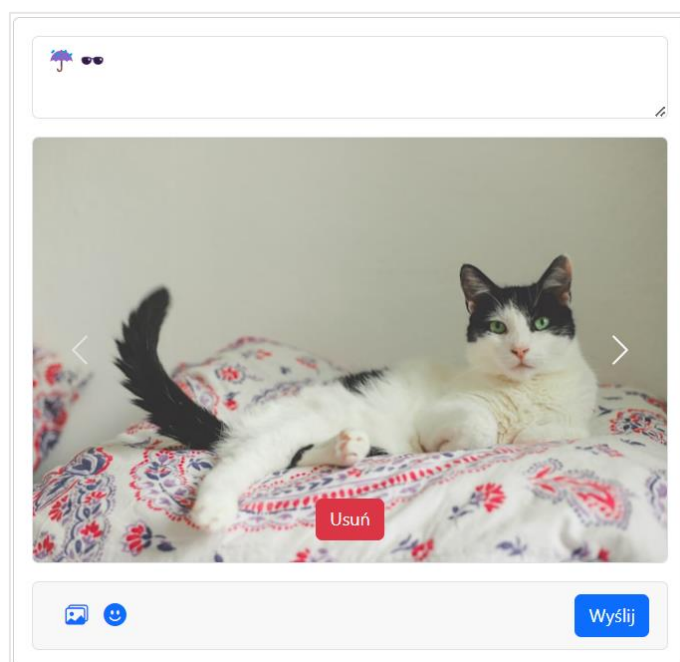


Zrzut ekranu 34 - Formularz tworzenia nowego posta

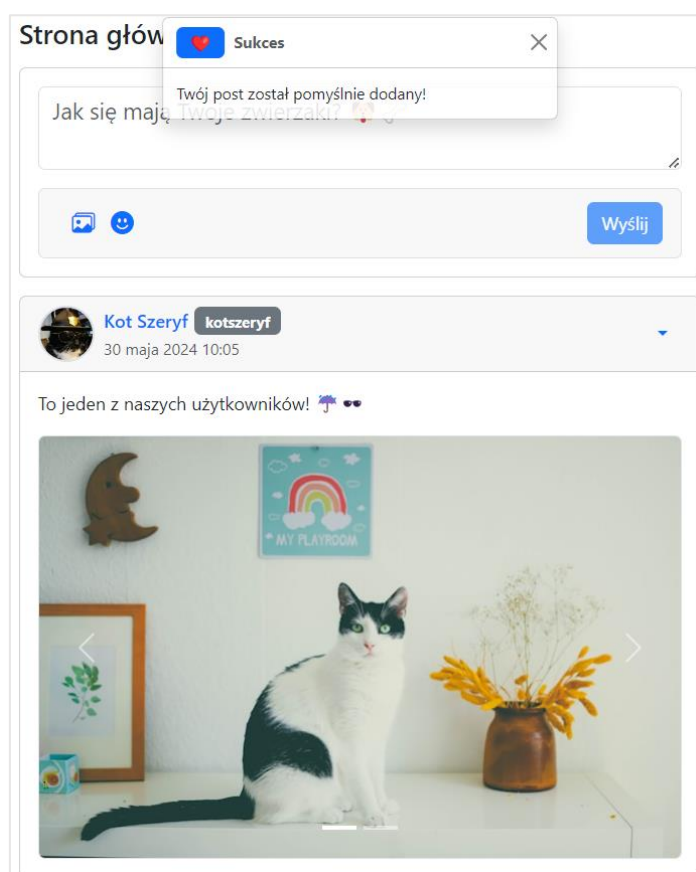


Zrzut ekranu 35 - Wybór emoji

Użytkownik może dodać do 5 załączników do swojego posta, które są wyświetlane pod polem treści wiadomości. Mechanizm pokazywania załączników korzysta z Bootstrap Carousels. Użytkownik ma możliwość usunięcia załącznika w dowolnym momencie.



Zrzut ekranu 36 - Widok załączników



Zrzut ekranu 37 - Powiadomienie po stworzeniu posta

Dodanie kolejnych załączników nie jest możliwe - osiągnięto maksymalną liczbę 5 załączników.



Wyślij

Zrzut ekranu 38 - Działanie walidacji po stronie Front-endu w przypadku próby wysłania zbyt dużej liczby załączników

Nieprawidłowy typ pliku onlypets.sql.



Wyślij

Zrzut ekranu 39 - Działanie walidacji po stronie Front-endu w przypadku próby wysłania pliku niebędącego obrazkiem

Rozmiar pliku pexels-eberhardgross-443446.jpg przekracza limit 2MB.



Wyślij

Zrzut ekranu 40 - Działanie walidacji po stronie Front-endu w przypadku próby wysłania zbyt dużego pliku

```
{!-- Accept any file with an image/* MIME type.
(Many mobile devices also let the user take a picture with the camera when this is used.) --}}
<input type="file" id="post-attachments-input" accept="image/*" multiple
class="d-none">
```

W kodzie JavaScript następuje nasłuchiwanie zmian na polu „postAttachmentsInput”. Kiedy użytkownik wybierze pliki, zostanie uruchomiona funkcja walidująca. Ta funkcja sprawdza, czy liczba załączników nie przekracza maksymalnej wartości określonej w zmiennej max. Jeśli liczba załączników jest już równa lub większa niż wartość max, zostanie wygenerowany błąd informujący użytkownika, że nie może dodać więcej załączników. Jeśli liczba załączników jest mniejsza niż maksymalna wartość, kod sprawdza typ każdego pliku. Jeśli plik nie jest obrazem (formaty: JPEG, PNG, JPG, GIF) lub jego rozmiar przekracza 2MB, zostanie wygenerowany odpowiedni komunikat o błędzie.

Zrzut ekranu 41 - Walidacja w kodzie HTML

```
const fileTypes = [
  "image/jpeg",
  "image/png",
  "image/jpg",
  "image/gif",
];

function validFileType(file) {
  return fileTypes.includes(file.type);
}
```

Zrzut ekranu 42 - Implementacja metody validFileType

```

postAttachmentsInput.addEventListener('change', (event) => {
    const files = event.target.files;
    const max = @json(env('MAX_POST_ATTACHMENTS'));

    for (const file of files) {
        if (attachments.length >= max) {
            errors.push(
                'Dodanie kolejnych załączników nie jest możliwe - osiągnięto maksymalną liczbę ${max} załączników.'
            );
            break;
        }

        if (!validFileType(file)) {
            errors.push('Nieprawidłowy typ pliku ${file.name}.');
            continue;
        }

        console.log(file.size);

        if (file.size > 2048) {
            errors.push('Rozmiar pliku ${file.name} przekracza limit 2MB.');

```

Zrzut ekranu 43 - Implementacja obsługi zdarzenia "change" oraz walidacja plików

W kodzie PHP, w kontrolerze obsługującym zapisywanie danych, są zdefiniowane reguły walidacji danych przekazywanych przez żądanie HTTP. Reguły te obejmują:

- Walidację załączników:
 - Walidacja dotyczy tablicy załączników.
 - Tablica załączników nie może przekraczać maksymalnej liczby załączników określonej w zmiennej środowiskowej MAX_POST_ATTACHMENTS.
 - Każdy załącznik musi być obrazem o dopuszczalnych formatach (JPEG, PNG, JPG, GIF) i maksymalnym rozmiarze 2048 KB.
- Walidację tekstu posta:
 - Jeśli nie dodano żadnych załączników, tekst posta jest wymagany.
 - Tekst posta może zawierać maksymalnie 255 znaków.

W przypadku naruszenia któregoś z tych warunków, zostaną wygenerowane odpowiednie komunikaty o błędach, które będą przekazywane użytkownikowi.

```

public function store(Request $request)
{
    $hasAttachments = $request->has('attachments');

    $rules = [
        'attachments' => 'array|max:' . env('MAX_POST_ATTACHMENTS'),
        'attachments.*' => 'image|mimes:jpeg,png,jpg,gif|max:2048',
        'text' => $hasAttachments ? '' : 'required' . 'string|max:255',
    ];

    $messages = [
        'attachments.max' => 'Możesz dodać maksymalnie ' . env('MAX_POST_ATTACHMENTS') . ' załączników.',
        'attachments.*.max' => 'Załącznik nr :index nie może być większy niż :max kilobajtów.',
        'attachments.*.mimes' => 'Załącznik nr :index musi być plikiem typu: :values.',
        'attachments.*.image' => 'Załącznik nr :index musi być obrazem.',
        'text.required' => 'Treść posta jest wymagana, jeśli nie dodano żadnych załączników.',
    ];

    $validator = Validator::make($request->all(), $rules, $messages);

    if ($validator->fails()) {
        return response()->json([
            'success' => false,
            'errors' => $validator->errors()
        ], 422);
    }
}

```

Zrzut ekranu 44 - Implementacja metody store (w tym walidacji) w kodzie PHP

6.5. Zarządzanie postami

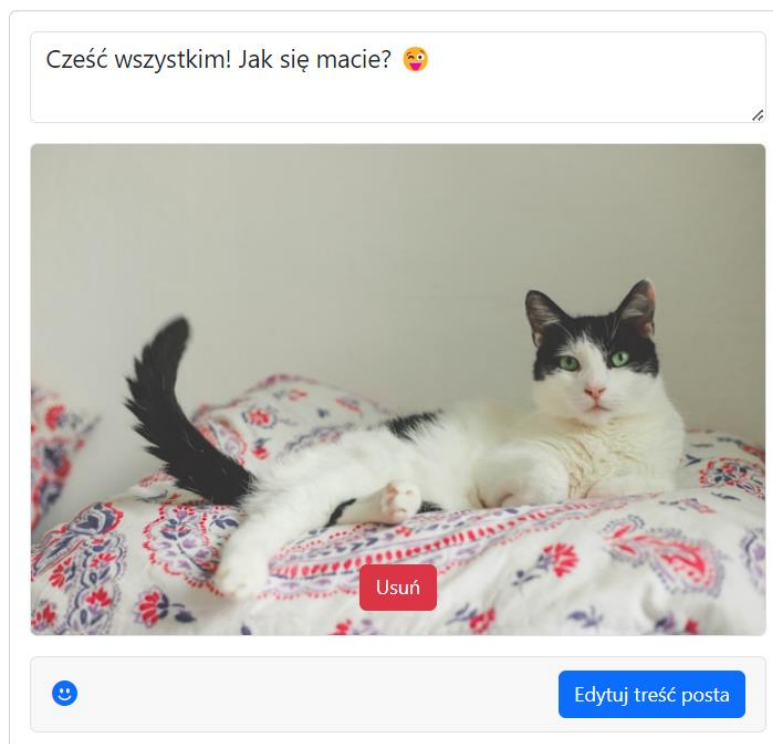
Użytkownicy mogą zarządzać swoimi postami, w tym tworzyć, przeglądać, edytować i usuwać je. Na karcie danego posta, w prawym górnym rogu, znajduje się strzałka, którą po kliknięciu można otworzyć rozwijane menu z opcjami "Edytuj post" oraz "Usuń post". Administratorzy aplikacji również posiadają opcję edycji oraz usunięcia postów użytkowników aplikacji.



Zrzut ekranu 45 - Rozwijane menu posta

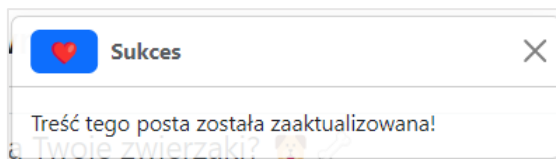
Po kliknięciu w opcję edycji, użytkownik jest przenoszony na nowy widok.

Edytowanie posta użytkownika Kotka Figa

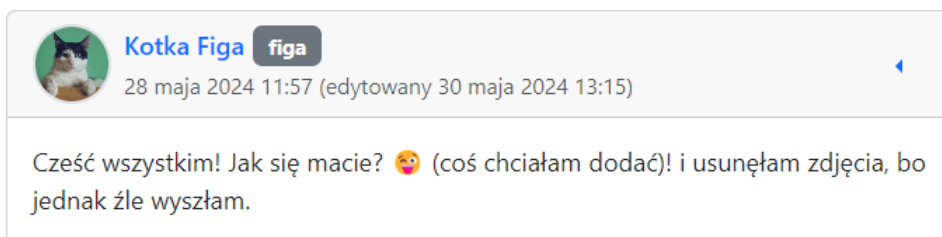


Zrzut ekranu 46 - Widok edycji posta

W górnej części karty edycji postu jest widoczne pole tekstowe, w którym możemy wprowadzić zmiany w treści postu. Poniżej znajdują się załączniki, które również jak post wykorzystują Bootstrap Carousels. Administrator lub autor posta ma możliwość usunięcia danego załącznika. Po zakończeniu edycji, aby zatwierdzić zmiany, użytkownik musi nacisnąć przycisk „Edytuj treść posta”.

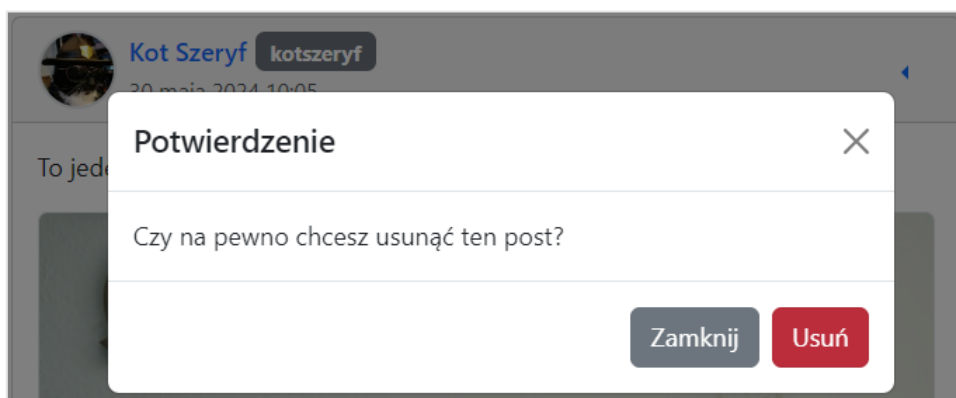


Zrzut ekranu 47 - Powiadomienie

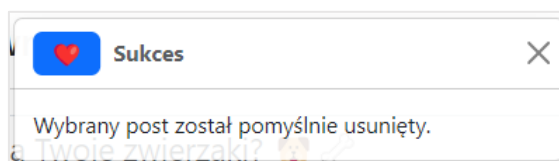


Zrzut ekranu 48 - Wygląd posta po edycji

Po kliknięciu w opcję usunięcia, użytkownikowi wyświetlane jest potwierdzenie. Użytkownik ma możliwość potwierdzić usunięcie posta, lub zrezygnować zamykając okno.



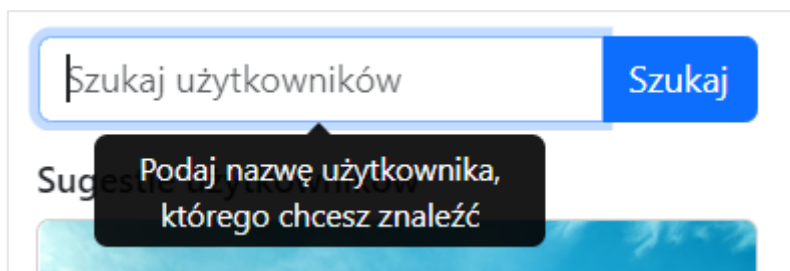
Zrzut ekranu 49 - Potwierdzenie usunięcia posta



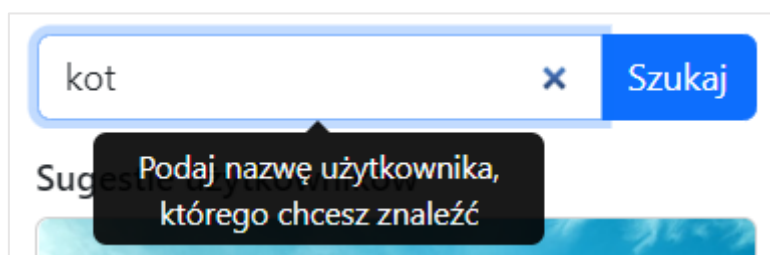
Zrzut ekranu 50 – Powiadomienie

6.6. Wyszukiwanie użytkowników

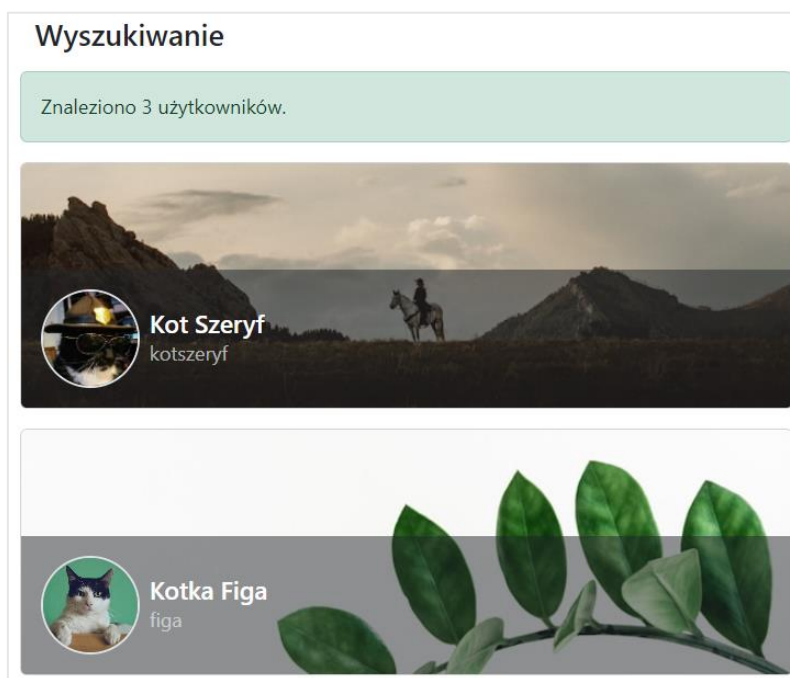
W prawym górnym rogu użytkownik znajdzie formularz służący do wyszukiwania innych użytkowników. Aby wyszukać użytkownika, należy wpisać jego nazwę.



Zrzut ekranu 51 - Formularz wyszukiwania

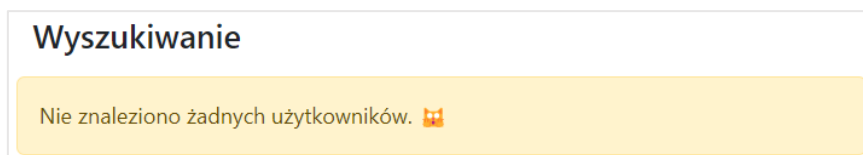


Zrzut ekranu 52 - Wyszukiwanie użytkowników z frazą "kot" w nazwie



Zrzut ekranu 53 - Wyniki wyszukiwania

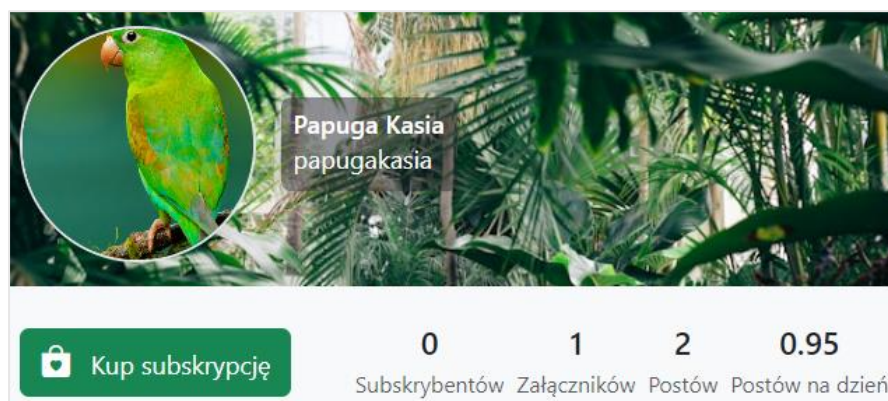
W przypadku nieudanego wyszukiwania, użytkownik zostanie odpowiednio powiadomiony.



Zrzut ekranu 54 - Nieudane wyszukiwanie

6.7. Subskrypcje

Każdy użytkownik ma możliwość wykupienia subskrypcji dla innego użytkownika. Aby to zrobić, należy wejść na profil danej osoby i nacisnąć przycisk "Kup subskrypcję".



Zrzut ekranu 55 - Karta użytkownika na jego profilu oraz przycisk "Kup subskrypcję"

Po naciśnięciu przycisku „Kup subskrypcję” użytkownik zostaje przekierowany do widoku kupna subskrypcji.

Kupno nowej subskrypcji

Informacja

Kupując subskrypcję dla użytkownika **Papuga Kasia**, uzyskasz dostęp do przeglądania jego postów oraz załączników.

80% płatności zostanie przekazane dla użytkownika, 15% zostanie przekazane na wsparcie [Schroniska dla zwierząt "Kundelek"](#) w Rzeszowie, a pozostałe 5% zostanie przeznaczone na obsługę serwisu.

Po zakończeniu subskrypcji będziesz musiał ją ponownie wykupić. W przypadku zbliżającego się końca subskrypcji otrzymasz powiadomienie.

Zakup subskrypcji na jeden miesiąc, będzie Cię kosztować 10 zł. Ta płatność jest jednorazowa.

Ustawienia subskrypcji

Długość subskrypcji
1 miesiąc

Cena subskrypcji
10 zł

Płatność

Imię i nazwisko na karcie
Kot Szeryf

Numer karty kredytowej
42424242424242

Data ważności
12/34

CVV
567

Przejdź do płatności

Zrzut ekranu 56 - Widok strony kupna subskrypcji

Zrzut ekranu 57 - Opcja zarządzania długością subskrypcji

Aplikacja informuje, co użytkownik uzyskuje w zamian za kupno subskrypcji. Użytkownik ma możliwość zarządzania długością subskrypcji, a w dolnej sekcji widoku zawarty jest formularz służący do wypełnienia danych dotyczących płatności. Wszystkie pola zawarte w tym formularzu są walidowane zarówno po stronie Front-end, jak i Back-end.

```
<input type="text" class="form-control @error('cc-name') is-invalid @enderror" id="cc-name"
name="cc-name" placeholder="Imię i nazwisko na karcie"
value="{{ old('cc-name', Auth::user()->display_name) }}" required maxlength="255">
```

Zrzut ekranu 58 - Walidacja po stronie Front-end imienia i nazwiska

```
<input type="text" class="form-control @error('cc-number') is-invalid @enderror"
id="cc-number" name="cc-number" placeholder="Numer karty kredytowej"
value="{{ old('cc-number', '4242424242424242') }}" required pattern="[0-9]{16}"
title="Numer karty kredytowej powinien składać się z 16 cyfr.">
```

Zrzut ekranu 59 - Walidacja po stronie Front-end numeru karty kredytowej

```
<input type="text" pattern="\d{2}/\d{2}"
class="form-control @error('cc-expiration') is-invalid @enderror" id="cc-expiration"
name="cc-expiration" placeholder="Data ważności"
value="{{ old('cc-expiration', '12/34') }}" required
pattern="^(0[1-9]|1[0-2])\./?([0-9]{4}|[0-9]{2})$"
title="Data ważności karty powinna być w formacie MM/YY lub MM/YYYY.">
```

Zrzut ekranu 60 - Walidacja po stronie Front-endu daty ważności karty kredytowej

```
<input type="number" min="100" max="999"
class="form-control @error('cc-cvv') is-invalid @enderror" id="cc-cvv" name="cc-cvv"
placeholder="CVV" value="{{ old('cc-cvv', '567') }}" required>
```

Zrzut ekranu 61 - Walidacja po stronie Front-endu daty ważności karty kredytowej

Formularz wykorzystuje różne metody walidacji w celu sprawdzenia poprawności wprowadzanych danych. Pola formularza mają atrybuty takie jak `required`, `maxlength`, `pattern` i `title`, które zapewniają podstawową walidację po stronie klienta. Na przykład, pole do wpisania numeru karty kredytowej musi zawierać dokładnie 16 cyfr, a data ważności musi być w formacie MM/YY lub MM/YYYY.

```

public function store(Request $request, User $user)
{
    /** @var \App\Models\User $subscriber */
    $subscriber = Auth::user();
    if ($subscriber->id == $user->id) {
        abort(403);
    }

    $request->validate([
        'cc-name' => 'required|string|max:255',
        'cc-number' => 'required|digits:16',
        'cc-expiration' => 'required|date_format:m/y',
        'cc-cvv' => 'required|digits:3',
        'length' => 'required|numeric|min:1'
    ]);

    Stripe::setApiKey(env('STRIPE_SECRET'));

    try {
        PaymentIntent::create([
            'amount' => 200,
            'currency' => 'pln',
            'payment_method' => 'pm_card_visa',
            'confirm' => true,
            'automatic_payment_methods' => [
                'enabled' => true,
                'allow_redirects' => 'never'
            ]
        ]);
        $length = (int) $request->input('length');
        $price = env('SUBSCRIPTION_MONTH_PRICE') * $length;

        $isNewSubscription = false;

        $existingSubscription = $subscriber->getSubscriptionForUser($user->id);
        if ($existingSubscription) {
            $sendDateTime = Carbon::parse($existingSubscription->end_at);
            $sendDateTime->addMonth($length);

            $existingSubscription->end_at = $sendDateTime->toDateTime();
            $existingSubscription->price = $price;
            $existingSubscription->show_notification = true;
            $existingSubscription->update();

            $toastMessage = 'Przedłużyłeś subskrypcję dla użytkownika <strong>' . $user->name . '</strong>!<br><br>Nowa data ważności: ';
        } else {
            $sendDateTime = Carbon::now()->addMonth($length);

            $subscriber->subscriptions()->create([
                'subscribed_user_id' => $user->id,
                'started_at' => now(),
                'end_at' => $sendDateTime->toDateTime(),
                'price' => $price,
            ]);

            $isNewSubscription = true;
            $toastMessage = 'Kupiłeś subskrypcję dla użytkownika <strong>' . $user->name . '</strong>!<br><br>Data ważności: ';
        }
        $toastMessage .= $sendDateTime->translatedFormat('d F Y, H:i') . '<br>Cena subskrypcji: <strong>' . $price . ' zł</strong>';

        return to_route('profile', $user)->with('successToast', Markdown::parse($toastMessage));
    } catch (Exception $e) {
        return redirect()
            ->back()
            ->withInput()
            ->with('paymentError', $e->getMessage());
    }
}

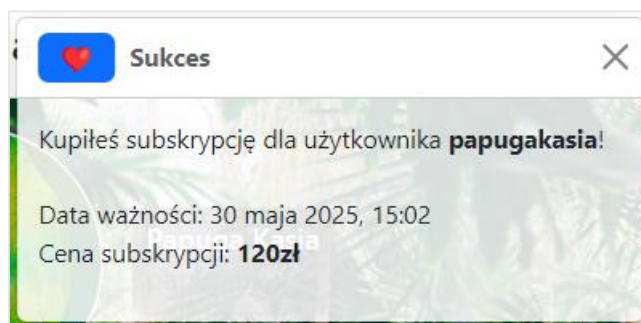
```

Zrzut ekranu 62 - Implementacja metody store w SubscriptionController

Metoda store obsługuje proces zakupu subskrypcji przez użytkownika. Najpierw pobiera aktualnie zalogowanego użytkownika i sprawdza, czy nie próbuje on subskrybować samego siebie. Następnie waliduje dane z formularza, takie jak imię i nazwisko na karcie, numer karty, data ważności i kod CVV.

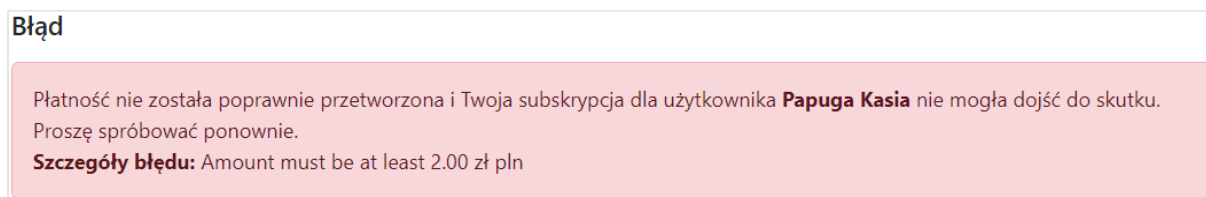
Po pomyślnej walidacji inicjuje transakcję płatności za pomocą Stripe. Jeśli użytkownik już posiada subskrypcję, przedłuża ją o podaną liczbę miesięcy. W przeciwnym razie tworzy nową subskrypcję. Po zakończeniu procesu subskrypcji wyświetla komunikat potwierdzający lub informujący o błędzie płatności.

W przypadku poprawnej finalizacji kupna subskrypcji, użytkownik zostaje powiadomiony.



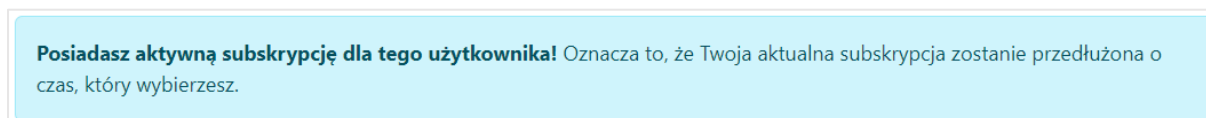
Zrzut ekranu 63 – Powiadomienie

W przeciwnym przypadku, zostanie poinformowany o błędzie.



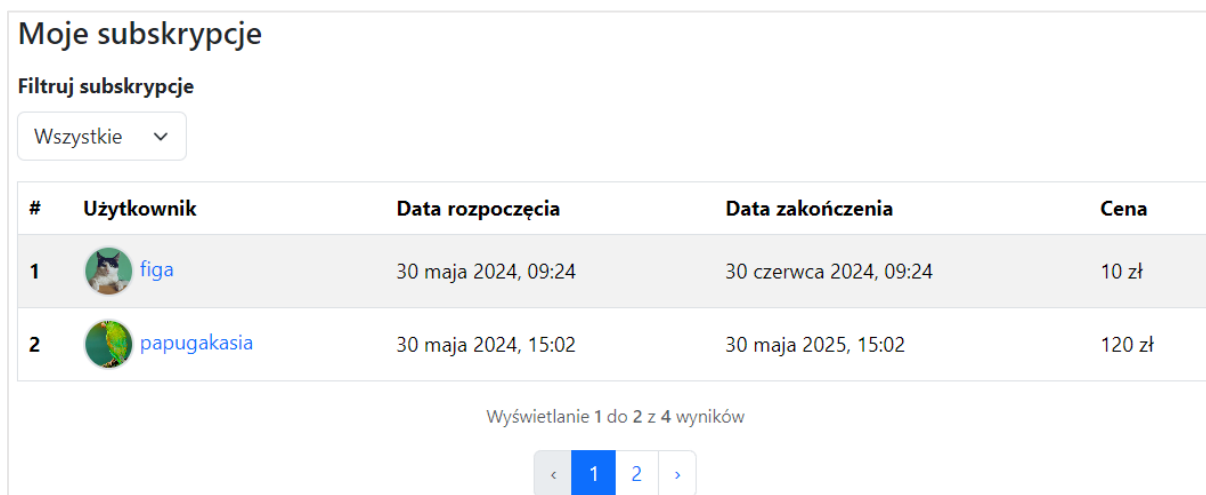
Zrzut ekranu 64 - Komunikat błędu

W przypadku, gdy użytkownik przedłuża subskrypcję, na stronie widoku kupna subskrypcji zostanie powiadomiony o tym, że przedłuża ważność subskrypcji.



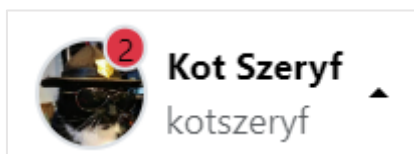
Zrzut ekranu 65 – Komunikat

Po wybraniu opcji „Moje subskrypcje” z głównego menu, użytkownik może zobaczyć informacje o swoich subskrypcjach dla innych użytkowników. Zawarte są tam takie informacje o tym dla którego użytkownika dana subskrypcja została kupiona, data rozpoczęcia, data zakończenia oraz ostatnia płatność. Istnieje również możliwość filtrowania subskrypcji.

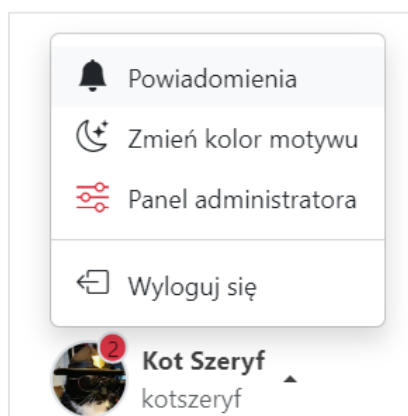


Zrzut ekranu 66 - Widok widoku subskrypcji użytkownika

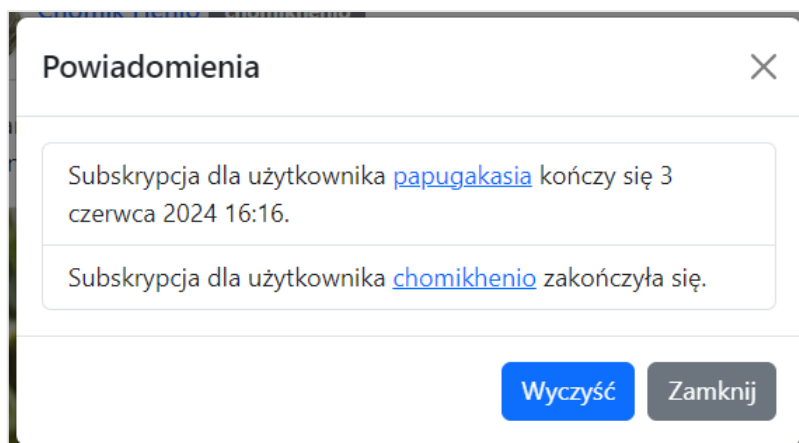
Aplikacja oferuje także opcję powiadamiania użytkowników o zbliżających się lub wygasłych terminach ważności subskrypcji. Gdy użytkownik posiada powiadomienia, w lewym dolnym rogu ekranu, nad awatarem wyświetlana jest czerwona kropka z liczbą powiadomień, które użytkownik posiada.



Zrzut ekranu 67 - Kropka powiadomień

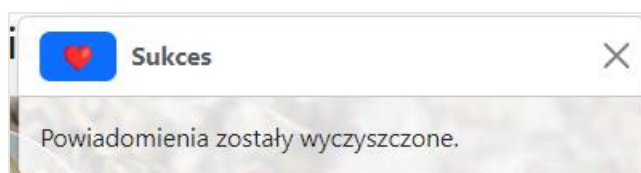


Zrzut ekranu 68 - Rozwijane menu



Zrzut ekranu 69 - Okienko powiadomień

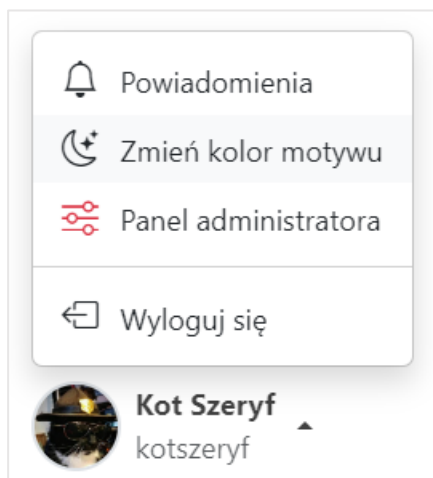
Użytkownik ma możliwość wyczyszczenia powiadomień – wtedy czerwona kropka nad awatarem znika, a te powiadomienia zostają „odczytane”.



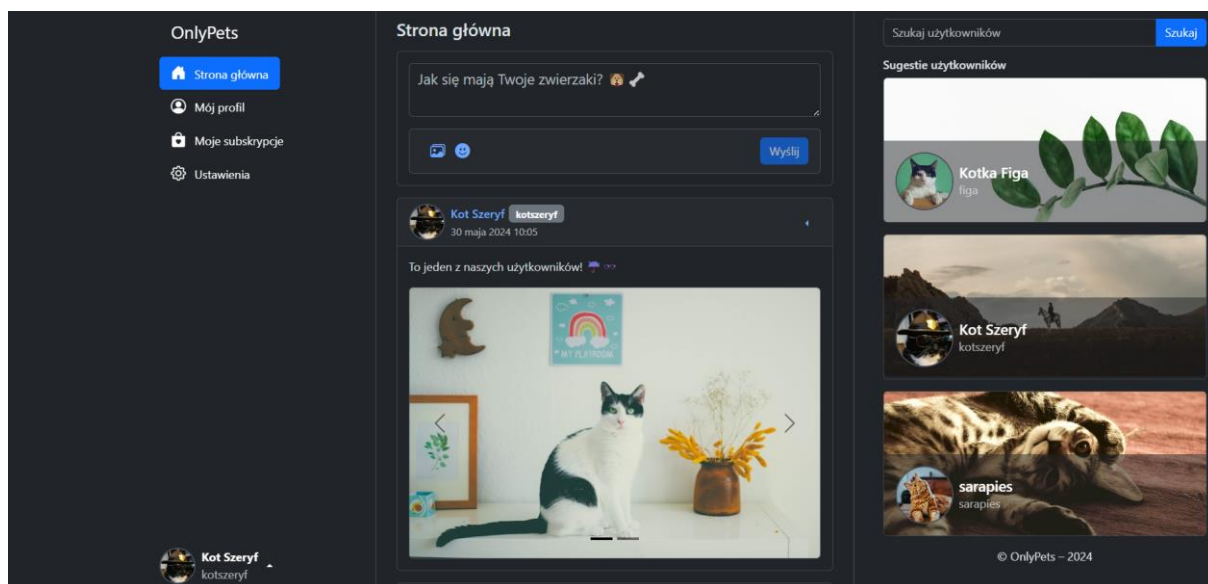
Zrzut ekranu 70 - Powiadomienie

6.8. Zmiana koloru motywu

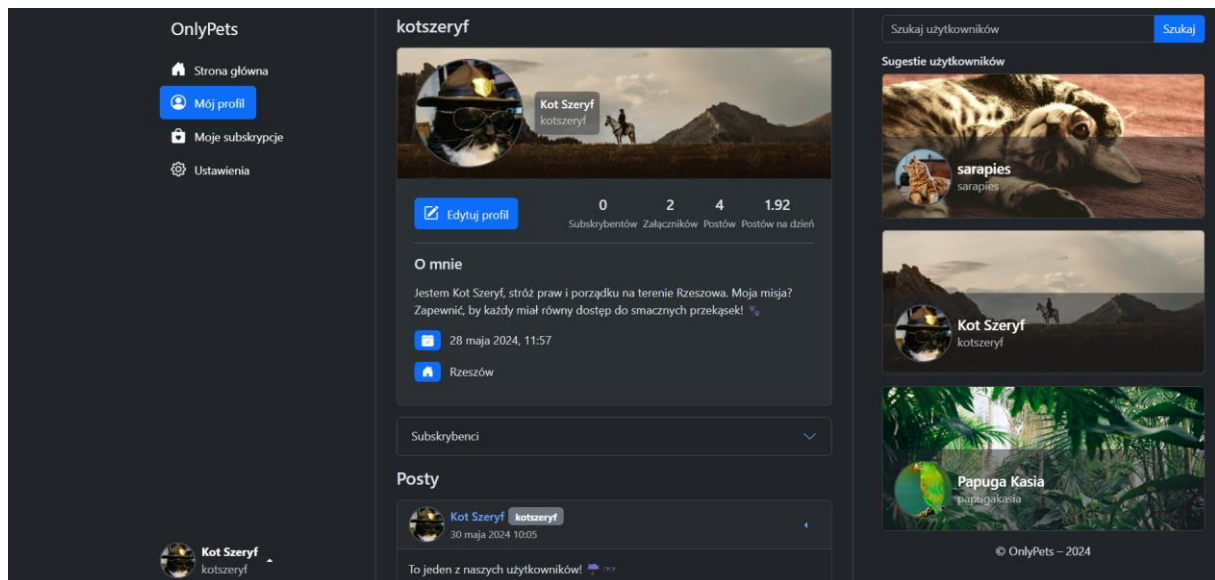
Użytkownicy mają możliwość zmiany motywu strony, która znajduje się w prawym dolnym rogu ekranu po rozwinięciu menu, lub w wersji mobilnej, po rozwinięciu górnego menu. Strona została dostosowana pod względem kolorystycznym do ciemnego motywu, co pozwala lepiej dopasować się do preferencji użytkowników.



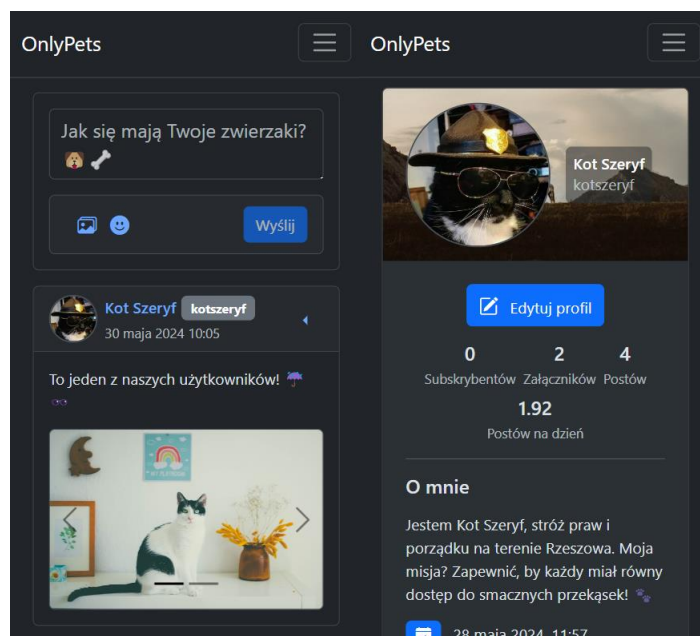
Zrzut ekranu 71 - Rozwijane menu użytkownika



Zrzut ekranu 72 - Strona główna w ciemnej wersji



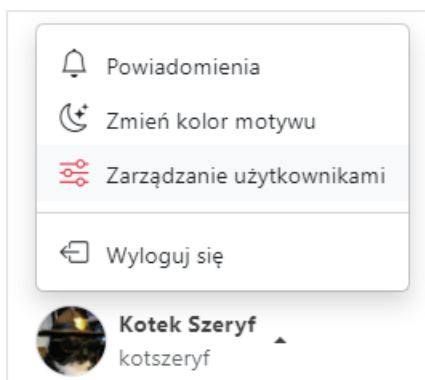
Zrzut ekranu 73 - Widok profilu użytkownika w ciemnej wersji



Zrzut ekranu 74 – Widoki aplikacji w ciemnej mobilnej wersji

6.9. Zarządzanie użytkownikami przez administratora

Administratorzy posiadają możliwość edycji danych użytkowników. Opcja przejścia do panelu administratora znajduje się w rozwijanym menu użytkownika.



Zrzut ekranu 75 - Rozwijane menu użytkownika

Po wybraniu tej opcji, administrator zostaje przeniesiony do widoku listy użytkowników, gdzie może wyszukiwać użytkowników za pomocą nazwy użytkownika lub adresu e-mail. Pod polem wyszukiwania znajduje się tabela zawierająca kluczowe informacje o użytkownikach, takie jak nazwa użytkownika, adres e-mail, data rejestracji, data ostatniej modyfikacji oraz opcje edycji i usuwania. Tabela korzysta z wbudowanej paginacji dostarczonej przez Laravel.

Wyszukaj użytkowników

Wyszukaj

#	Użytkownik	Adres e-mail	Data rejestracji	Data ostatniej zmiany		
1	kotszeryf	kotszeryf@email.com	31 maja 2024, 08:34	05 czerwca 2024, 19:42	<button>Edytuj</button>	<button>Usuń</button>
2	chomikhenio	chomikhenio@email.com	31 maja 2024, 08:34	05 czerwca 2024, 19:41	<button>Edytuj</button>	<button>Usuń</button>
3	kroliczekbaska	kroliczekbaska@email.com	31 maja 2024, 08:34	05 czerwca 2024, 19:26	<button>Edytuj</button>	<button>Usuń</button>

Wyświetlanie 1 do 3 z 5 wyników

< 1 2 >

Zrzut ekranu 76 - Lista użytkowników

Po wybraniu opcji edycji, administrator zostaje przeniesiony do widoku edycji użytkownika, w którym może dokonać edycji takich pól jak: nazwa, nazwa wyświetlana, hasło, adres strony internetowej użytkownika, bio, lokalizacja oraz awataru i zdjęcia w tle użytkownika.

Strona główna / Edycja użytkownika kroliczekbaska

Informacje osobiste

Nazwa

kroliczekbaska

Nazwa wyświetlana

Króliczek Baśka

Bio i kontakt

Bio

Jestem puszystym królikiem, który uwielbia skakać po ogrodzie i jeść świeże warzywa.

Miejscowość

Gdańsk

Adres strony internetowej

Edytuj użytkownika

Zmiana hasła

Nowe hasło


Powtórz nowe hasło

Awatar i zdjęcie w tle

Wybierz plik

Nie wybrano pliku

Usuń aktualny awatar

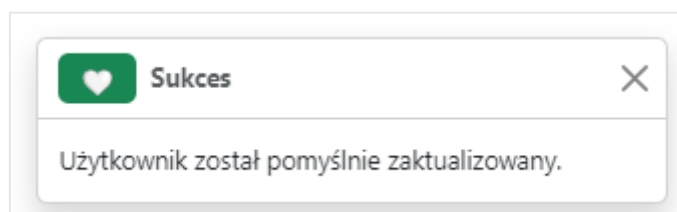


Awatar i zdjęcie w tle

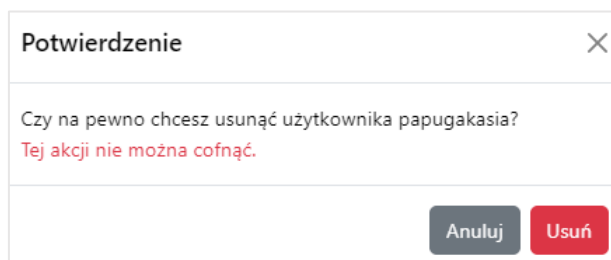
Wybierz plik

Nie wybrano pliku

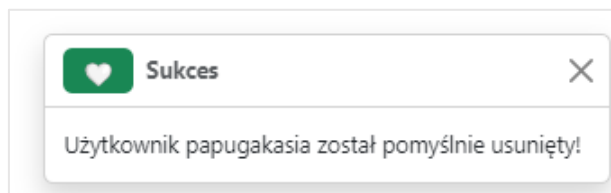
Aby zatwierdzić wprowadzone dane, należy nacisnąć przycisk „Edytuj użytkownika”.



Po wybraniu opcji usunięcia użytkownika, wyświetlane jest okno potwierdzenia.



Zrzut ekranu 77 – Potwierdzenie



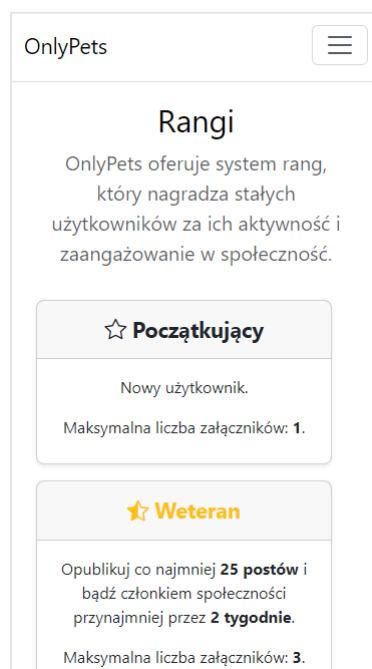
Zrzut ekranu 78 – Powiadomienie

6.10. Rangi użytkowników

Aplikacja wprowadza system rang, który nagradza stałych użytkowników za ich aktywność i zaangażowanie w społeczność. Aby dowiedzieć się więcej o systemie rang oraz korzyściach płynących z posiadania poszczególnych rang, użytkownicy mogą kliknąć przycisk „Rangi” w głównym menu aplikacji.

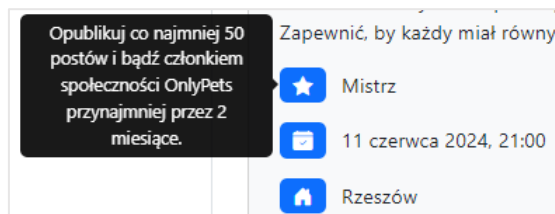


Zrzut ekranu 79 - Widok rang



Zrzut ekranu 80 – Widok rang w wersji mobilnej

Na profilu każdego użytkownika jest widoczna informacja, jaką rangę posiada dany użytkownik. Po najejchaniu na ikonę rangi, wyświetlany jest jej krótki opis.



Zrzut ekranu 81 - Informacja o randze użytkownika

```
public static function calculateRank(User $user): self
{
    $registeredAt = $user->created_at;
    $now = Carbon::now();
    $diffInDays = $now->diffInDays($registeredAt);

    if ($diffInDays < 14) {
        return self::Novice;
    } elseif ($diffInDays >= 14 && $diffInDays < 60) {
        return $user->posts()->count() >= 25 ? self::Intermediate : self::Novice;
    } else {
        return $user->posts()->count() >= 50 ? self::Advanced : self::Intermediate;
    }
}
```

Zrzut ekranu 82 - Implementacja funkcji wyznaczającej rangę użytkownika

Metoda `calculateRank` oblicza rangę użytkownika na podstawie jego aktywności i czasu od rejestracji. Najpierw pobierana jest data rejestracji użytkownika i bieżąca data. Następnie obliczana jest różnica w dniach między tymi datami.

Jeśli użytkownik jest zarejestrowany krócej niż 14 dni, przyznawana jest mu ranga "Początkujący". Jeśli użytkownik jest zarejestrowany od 14 do 60 dni i opublikował co najmniej 25 postów, otrzymuje rangę "Weteran". W przeciwnym razie pozostaje na randze "Początkujący".

Jeżeli użytkownik jest zarejestrowany dłużej niż 60 dni i opublikował co najmniej 50 postów, przyznawana jest mu ranga "Mistrz". W przeciwnym razie otrzymuje rangę "Weteran".

```
public function getRank(): RankEnum
{
    return RankEnum::calculateRank($this);
}
```

Zrzut ekranu 83 - Implementacja metody zwracającej rangę użytkownika w modelu User

W zależności od posiadanej rangi, użytkownicy mają ograniczoną liczbę załączników, które mogą dodać do postów. Użytkownicy z rangą „Początkujący” mogą zamieścić maksymalnie 1 załącznik. Użytkownicy z rangą „Weteran” mogą zamieścić maksymalnie 3 załączniki. Natomiast użytkownicy z rangą „Mistrz” mogą zamieścić maksymalnie 6 załączników.

Gdy użytkownik spróbuje wystać więcej załączników, niż pozwala na to jego ranga, wyświetlany jest komunikat.

Jak się mają Twoje zwierzaki? 🐾🦴

Usuń

Dodanie kolejnych załączników nie jest możliwe - osiągnięto maksymalną liczbę 6 załączników.

Wyślij

Zrzut ekranu 84 – Komunikat

Innymi benefitami płynącymi z posiadania wyższych rang są zniżki na cenę miesięcznej subskrypcji. Użytkownicy posiadający rangę "Weteran" mogą liczyć na zniżkę 30% na cenę miesięcznej subskrypcji, zaś użytkownicy z rangą "Mistrz" mogą liczyć aż na 50% zniżki.

Ustawienia subskrypcji

Długość subskrypcji
1 miesiąc

Cena subskrypcji
~~10.00 zł~~ 5.00 zł

Posiadasz aktywną subskrypcję dla tego użytkownika! Oznacza to, że Twoja aktualna subskrypcja zostanie przedłużona o czas, który wybierzesz.

Płatność

Imię i nazwisko na karcie
Kot Szeryf

Numer karty kredytowej
4242424242424242

Data ważności
12/34

CVV
567

Twoja ranga to **Mistrz**, więc otrzymujesz zniżkę **50%** na miesięczną subskrypcję!

Zrzut ekranu 85 - Informacja o zniżkach

```

public function getSubscriptionDiscount(): int
{
    return match ($this) {
        self::Novice => 0,
        self::Intermediate => 30,
        self::Advanced => 50,
    };
}

```

Zrzut ekranu 86 - Implementacja funkcji zwracającej zniżkę

```

$length = (int) $request->input('length');
$price = env('SUBSCRIPTION_MONTH_PRICE') * $length;

$discount = $subscriber->getRank()->getSubscriptionDiscount();
if ($discount > 0) {
    $discountedPrice = $price - ($price * ($discount / 100));
    $price = $discountedPrice;
}

```



Zrzut ekranu 87 - Obliczanie finalnej ceny za subskrypcję

Oprócz wspomnianych korzyści, użytkownicy z rangą "Weteran" oraz "Mistrz" mają możliwość włączenia autoodnawiania subskrypcji. W zakładce "Subskrypcje" znajduje się przełącznik, który umożliwia zarządzanie tą opcją dla każdej subskrypcji.

Strona główna / **Moje subskrypcje**

Filtruj subskrypcje

Wszystkie ▾

#	Użytkownik	Data rozpoczęcia	Data zakończenia	Cena	Autoodnawianie
1	 kroliczekbaska	11 czerwca 2024, 22:51	11 lipca 2024, 22:51	5 zł	<input checked="" type="checkbox"/>
2	 papugakasia	05 czerwca 2024, 13:26	12 czerwca 2024, 12:26	5 zł	<input type="checkbox"/>

Zrzut ekranu 88 - Widok subskrypcji użytkownika

```

<td class="align-middle">
    @if ($subscription->subscriber->getRank()->canAutoRenewSubscription())
        <div class="form-check form-switch">
            @if ($subscription->isExpired())
                <input class="form-check-input" type="checkbox" role="switch" disabled>
            @else
                <input class="form-check-input" type="checkbox" role="switch"
                    id="autoRenewSwitch-{{ $subscription->id }}"
                    {{ $subscription->auto_renew ? 'checked' : '' }}>
            @endif
        </div>
    @else
        <div class="form-check form-switch d-inline-block" tabindex="0"
            data-bs-toggle="tooltip"
            title="Twoja ranga nie pozwala na autoodnawianie subskrypcji!">
            <input class="form-check-input" type="checkbox" role="switch" disabled>
        </div>
    @endif
</td>

```

Zrzut ekranu 89 - Kod HTML kolumny zawierającej przełączniki

Kod HTML wyświetla przełącznik autoodnawiania subskrypcji. Sprawdza, czy użytkownik ma rangę pozwalającą na autoodnawianie subskrypcji oraz czy subskrypcja nie wygasta. Jeśli oba warunki są spełnione, przełącznik jest aktywny i może być włączony lub wyłączony w zależności od stanu autoodnawiania. W przeciwnym razie, przełącznik jest wyłączony, a w przypadku braku odpowiedniej rangi, wyświetlana jest informacja w formie tooltipa.

```

<script>
    const autoRenewSwitches = document.querySelectorAll('[id^="autoRenewSwitch-"]');

    autoRenewSwitches.forEach(autoRenewSwitch => {
        autoRenewSwitch.addEventListener('change', function() {
            const subscriptionId = this.id.split('-').pop();
            const isChecked = this.checked ? true : false;

            fetch("{ route('subscriptions.switch.auto.renew') }", {
                method: 'post',
                headers: {
                    'Content-Type': 'application/json',
                    'X-CSRF-TOKEN': '{{ csrf_token() }}'
                },
                body: JSON.stringify({
                    subscription_id: subscriptionId,
                    auto_renew: isChecked
                })
            })
                .then(response => response.json())
                .then(data => {
                    if (data.errors) {
                        const errorMessages = Object.values(data.errors).flat().join('\n');
                        console.error(errorMessages);
                    }
                })
                .catch(error => {
                    console.error(error.message);
                });
        });
    });
</script>

```

Zrzut ekranu 90 - Kod JavaScript obsługujący przełączniki

Kod w JavaScript dodaje nasłuchiwanie na zmianę stanu przełączników autoodnawiania subskrypcji. Dla każdego przełącznika (aktualnej subskrypcji) po zmianie jego stanu, wysyłane jest żądanie POST do serwera z aktualnym stanem autoodnawiania. Żądanie zawiera identyfikator subskrypcji i nowy stan autoodnawiania w formacie JSON, a także token CSRF dla zabezpieczenia. Odpowiedź z serwera jest sprawdzana pod kątem błędów, które są wyświetlane w konsoli, jeśli wystąpią.

```

public function switchAutoRenew(Request $request)
{
    $validator = Validator::make($request->all(), [
        'subscription_id' => 'required|integer|exists:subscriptions,id',
        'auto_renew' => 'required|boolean'
    ]);

    if ($validator->fails()) {
        return response()->json(['errors' => $validator->errors()], 400);
    }

    /** @var \App\Models\User $user */
    $user = Auth::user();
    $subscription = Subscription::findOrFail($request->input('subscription_id'));

    if (!$user || $subscription->subscriber_user_id != $user->id || !$user->getRank()->canAutoRenewSubscription()) {
        return response()->json(null, 403);
    }

    $subscription->auto_renew = $request->input('auto_renew');
    if ($subscription->save()) {
        return response()->json();
    } else {
        return response()->json(['errors' => ['save' => 'Nie udało się zapisać zmian w subskrypcji.']], 500);
    }
}

```

Zrzut ekranu 91 - Kod PHP obsługujący żądanie POST

Metoda `switchAutoRenew` waliduje dane wejściowe żądania dotyczące zmiany stanu autoodnawiania subskrypcji. Sprawdza, czy użytkownik jest zalogowany, czy subskrypcja należy do niego oraz czy ma prawo do zarządzania autoodnawianiem. Jeśli którykolwiek z tych warunków nie jest spełniony, zwraca odpowiedź z błędem. W przeciwnym razie aktualizuje stan `auto_renew` subskrypcji i zapisuje zmiany.

Algorytm autoodnawiania subskrypcji wykorzystuje prace i kolejki w Laravel. Dla każdej subskrypcji tworzona jest nowa praca, która zostaje uruchomiona w momencie wygaśnięcia subskrypcji.

```
$diffInSeconds = now()->diffInSeconds($endDateTime);
$job = (new RenewSubscriptionJob($subscription))->delay(now()->addSeconds($diffInSeconds));
$jobId = dispatchId($job);

$oldJobId = $subscription->job_id;
$subscription->job_id = $jobId;
$subscription->update();

if ($oldJobId) {
    DB::table('jobs')->where('id', $oldJobId)->delete();
}
```

Zrzut ekranu 92 - Implementacja tworzenia pracy dla subskrypcji w metodzie `store` kontrolera `SubscriptionController`

```
<?php

namespace App\Http\Controllers;

use Illuminate\Container\Container;
use Illuminate\Contracts\Bus\Dispatcher;
use Illuminate\Contracts\Queue\ShouldQueue;

// https://github.com/arispatri/laravel-dispatch-id
class RenewSubscriptionController extends Controller
{
    public static function dispatch(ShouldQueue $job): int
    {
        return static::dispatcher()->dispatch($job);
    }

    private static function dispatcher()
    {
        return Container::getInstance()->make(Dispatcher::class);
    }
}
```

Zrzut ekranu 93 - Implementacja `RenewSubscriptionController`

```

public function handle(): void
{
    $subscription = $this->subscription;

    // aktualny job za chwilę zostanie usunięty, więc subskrypcja musi "rozłączyć" job_id,
    // ze względu na ograniczenia klucza obcego.

    $subscription->job_id = null;
    $subscription->save();

    $subscriber = $subscription->subscriber;
    if ($subscription->auto_renew && $subscriber->getRank()->canAutoRenewSubscription()) {
        $price = env('SUBSCRIPTION_MONTH_PRICE') * 1;

        $discount = $subscriber->getRank()->getSubscriptionDiscount();
        if ($discount > 0) {
            $discountedPrice = $price - ($price * ($discount / 100));
            $price = $discountedPrice;
        }

        $sendDateTime = Carbon::now()->addMonth(1);
        // $sendDateTime = Carbon::now()->addSecond(10);

        $subscription->end_at = $sendDateTime->toDateTime();
        $diffInSeconds = now()->diffInSeconds($sendDateTime);

        $job = (new RenewSubscriptionJob($subscription))->delay(now()->addSeconds($diffInSeconds));
        $jobId = dispatchId($job);

        $subscription->job_id = $jobId;
        $subscription->price = $price;
        $subscription->save();
    }

    // w przeciwnym wypadku subskrypcja nie zostanie przedłużona i wygaśnie
}

```

Zrzut ekranu 94 - Implementacja metody handle w RenewSubscriptionJob

Metoda `handle` zajmuje się autoodnawianiem subskrypcji. Najpierw ustawia `job_id` subskrypcji na `null`, aby usunąć powiązanie z bieżącą pracą. Następnie, jeśli autoodnawianie jest włączone i użytkownik ma odpowiednią rangę, oblicza cenę subskrypcji, uwzględniając ewentualny rabat. Następnie ustala nową datę zakończenia subskrypcji i tworzy nową pracę `RenewSubscriptionJob`, która zostanie uruchomiona w momencie wygaśnięcia subskrypcji. Aktualizuje `job_id` i cenę subskrypcji oraz zapisuje zmiany w bazie danych. Jeśli warunki autoodnawiania nie są spełnione, subskrypcja wygaśnie.

Aby zapewnić poprawne działanie mechanizmu autoodnawiania subskrypcji w aplikacji, należy przeprowadzić poniższe kroki:

- **Restart kolejki:** Wywołanie polecenia `php artisan queue:restart`, aby zresetować wszelkie prace w kolejce. Ten krok zapewnia aktualność środowiska i przygotowuje je do obsługi nowych zadań.
- **Uruchomienie procesu obsługi kolejki:** Uruchomienie procesu obsługi kolejki za pomocą polecenia `php artisan queue:work`. Proces ten stale monitoruje pojawianie się nowych zadań w kolejce i wykonuje je zgodnie z ich pojawieniem się. Jest to kluczowy proces dla sprawnego działania mechanizmu autoodnawiania subskrypcji, ponieważ gwarantuje on, że zadania są realizowane w odpowiednim czasie.

7. Źródła

Aplikacja do stworzenia przykładowych danych używa darmowych ilustracji pochodzących ze strony <https://unsplash.com/>.