

| AI1 | Dokumentacja projektu |
|-----------------------|--|
| Autor | Jakub Jakubowski, 125125 |
| Kierunek, rok | Informatyka, II rok, st. stacjonarne (3,5-l) |
| Temat projektu | <i>Zoo – rezerwacja wizyt grupowych</i> |

Spis treści

| | |
|--|----|
| Wstęp | 3 |
| Narzędzia i technologie | 4 |
| Baza danych..... | 5 |
| GUI | 6 |
| Desktopowe | 6 |
| Mobilne | 10 |
| Uruchomienie aplikacji..... | 14 |
| Funkcjonalności aplikacji | 18 |
| Logowanie | 19 |
| Rejestracja | 20 |
| Rezerwacja biletów przez gościa | 24 |
| Rezerwacja wizyt grupowych | 29 |
| Zarządzanie rezerwacjami użytkownika | 31 |
| Zasoby zalogowanego użytkownika | 32 |
| Zarządzanie użytkownikami..... | 35 |
| Zarządzanie rodzajami biletów | 39 |
| Zarządzanie rezerwacjami..... | 42 |
| Zarządzanie biletami rezerwacji..... | 46 |
| Statystyki..... | 48 |

Wstęp

Tematem projektu jest utworzenie strony w tematyce rezerwacji biletów do zoo.

Niezałogowany użytkownik może założyć nowe konto, przeglądać udostępnione przez stronę internetową zasoby, a także złożyć rezerwację do zoo, która jest ograniczona ilościowo. Załogowany użytkownik ma dostęp do zarządzania swoim kontem, zasobami oraz rezerwacji biletów grupowych i wyświetlania swoich rezerwacji. Funkcjonalnością wspólną dla obydwu wyżej wspomnianych klientów jest formularz kontaktowy. Administrator posiada możliwość zarządzania zarejestrowanymi użytkownikami, rezerwacjami, a także dodawaniem nowych biletów

Projekt znajduje się pod linkiem:

<https://github.com/Rolaski/Zooland>

Narzędzia i technologie

Wersja PHP (w CMD wpisać `php -v`)

- PHP 8.2.12 (cli) (built: Oct 24 2023 21:15:15) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.12, Copyright (c) Zend Technologies

Wersja Laravel'a (w terminalu command prompt wpisać polecenie `php artisan --version`)

- Laravel Framework 11.7.0

Wersja XAMPP: 3.3.0

Wersja Tailwind (w terminalu wpisać poleceni `npm view tailwindcss version`): 3.4.3

Wersja Node.js (w terminalu wpisać poleceni `node -v`): v22.2.0

Wersja Node Package Manager (w terminalu wpisać poleceni `npm -v`): 10.7.0

Wersja Visual Studio Code: 1.89.1 (user setup)

Dokumentacje i technologie:

Laravel: <https://laravel.com/docs/11.x/>

PHP: <https://www.php.net/manual/en/>

Baza danych: <https://docs.phpmyadmin.net/en/latest/>

Tailwind: <https://tailwindcss.com/docs/installation>

Node Package Manager: <https://docs.npmjs.com/>

JavaScript: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

HTML5: <https://developer.mozilla.org/en-US/docs/Glossary/HTML5>

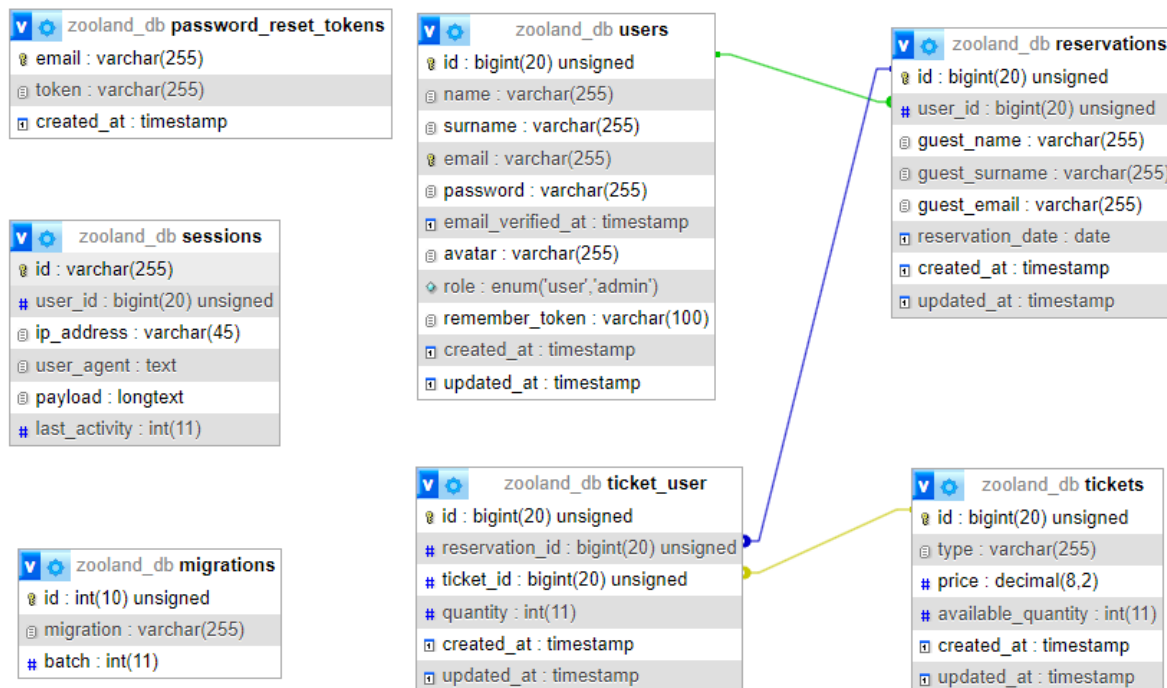
Visual Studio Code: <https://code.visualstudio.com/>

XAMPP: <https://www.apachefriends.org/pl/index.html>

Composer Setup: <https://getcomposer.org/Composer-Setup.exe>

Baza danych

Diagram ERD wygenerowany za pomocą phpMyAdmin



Rysunek 1 - diagram ERD

Baza danych zawiera takie tabele jak Tickets, która zawiera informacje na temat rodzajów biletów, ich ceny, które mogą być rezerwowane przez klientów.

Ticket_user trzyma informacje o tym jakie konkretnie bilety i ich ilość została zarezerwowana dla danej rezerwacji.

Reservations to tabela w której znajdują się podstawowe informacje o rezerwacji, czyli data oraz dane osoby powiązanej z rezerwacją, w przypadku niezalogowanego użytkownika wykorzystywane są pola gościa (guest_name, guest_surname, guest_email).

User zawiera dane użytkownika zalogowanego, który utworzył konto.

Tabele sessions, password_reset_tokens, migrations są automatycznie tworzone przez framework Laravel.

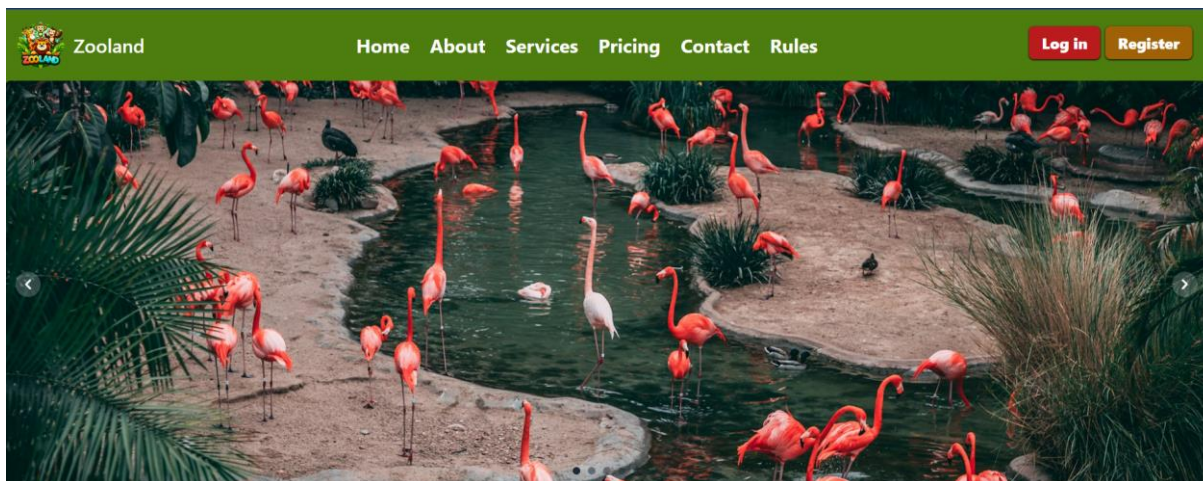
Relacje:

- Tabela users do tabeli reservations – jeden do wielu
- Tabela reservations do tabeli ticket_user – jeden do wielu
- Tabela tickets do tabeli ticket_user – jeden do wielu
- Tabela ticket_user do tabeli tickets – wiele do jednego

GUI

Desktopowe

Strona główna aplikacji:



Rysunek 2 – karuzela zdjęć zoo



Rysunek 3 - oferta biletów

Immerse yourself in the unique world of nature

At Zooland, you'll discover that we offer an experience that goes beyond the typical zoo visit. Here, our mission is to immerse you in a world of wonder and learning, where every corner holds a new adventure.

One of our standout features is the serene Japanese Garden, a tranquil oasis within the zoo. This beautifully landscaped garden offers a peaceful retreat to unwind and connect with nature. Stroll through the flora, listen to the soothing water features, and enjoy a moment of calm in this picturesque setting. The Japanese Garden at Zooland exemplifies our commitment to creating a multifaceted experience for our guests.



Discover the wonders of the deep blue sea

At Zooland, the adventure continues beneath the waves in our spectacular Oceanarium. Dive into an underwater world where you'll encounter a mesmerizing variety of marine life. From colorful coral reefs to majestic sea creatures, each exhibit is designed to educate and inspire.

One of our highlights is the interactive touch pool, where you can get up close and personal with gentle sea creatures. Learn about their habitats and behaviors from our knowledgeable staff, and gain a deeper appreciation for the mysteries of the ocean. The Oceanarium at Zooland is a testament to our dedication to providing an enriching and immersive experience for all our visitors.

Experience thrilling adventures at our Rope Park

At Zooland, the excitement doesn't end on the ground. Venture into the treetops and embark on an exhilarating journey through our Rope Park. Challenge yourself on a variety of obstacles, from suspended bridges to zip lines, as you navigate the forest canopy like never before.

Our Rope Park offers an adrenaline-pumping experience for adventurers of all ages. Whether you're a seasoned thrill-seeker or trying something new, our professionally designed courses ensure a safe and unforgettable adventure. Soar through the trees and conquer your fears at the Rope Park, only at Zooland.



Rysunek 4 - informacja o atrakcjach

Reserve Your Visit to Our Zoo

Don't wait, book your tickets today! Immerse yourself in the fascinating world of our zoo. Spaces are limited, so don't miss your chance for unforgettable experiences. See you at our zoo!

Select Date

dd.mm.2024

Standard - \$45.00

—

0

+

Discount - \$29.99

—

0

+

Senior - \$35.00

—

0

+

VIP - \$99.99

—

0

+

Next

You can book a maximum of 5 tickets for each category. If you want to book a group visit, create an account!

Contact Us

Have a question about our animals? Want to share your experience or provide feedback? Interested in learning more about our educational programs and events? Reach out to us!

Your email

name@email.com

Subject

Let us know how we can help you

Your message

Leave a comment...

Send message

Rysunek 5 - rezerwacja biletów oraz kontakt



Zooland

Opening hours

Mon - Fri: 9:00 AM - 6:00 PM
Sat: 9:00 AM - 5:00 PM
Sun: 10:00 AM - 4:00 PM

We invite you all year round except:

January 1 - closed
Easter Sunday - closed
November 1 - closed
December 24, 25 - closed

Address

Poland, 35-399 Rzeszów
Krakowska 194th St.


Contact

+48 697 931 854
+48 984 245 787
contact@zooland.pl

© 2024 Zooland. All Rights Reserved.

Rysunek 6 - stopka, informacje o zoo

Rezerwacja biletów dla gościa:



Zooland

[Home](#) [About](#) [Services](#) [Pricing](#) [Contact](#) [Rules](#)

[Log in](#) [Register](#)

First name

John

Last name

Doe

Email address

john.doe@company.com

☒ I agree with the [terms and conditions](#).

Visit Date: 2024-05-31

Standard: 2 ticket(s)

Discount: 1 ticket(s)

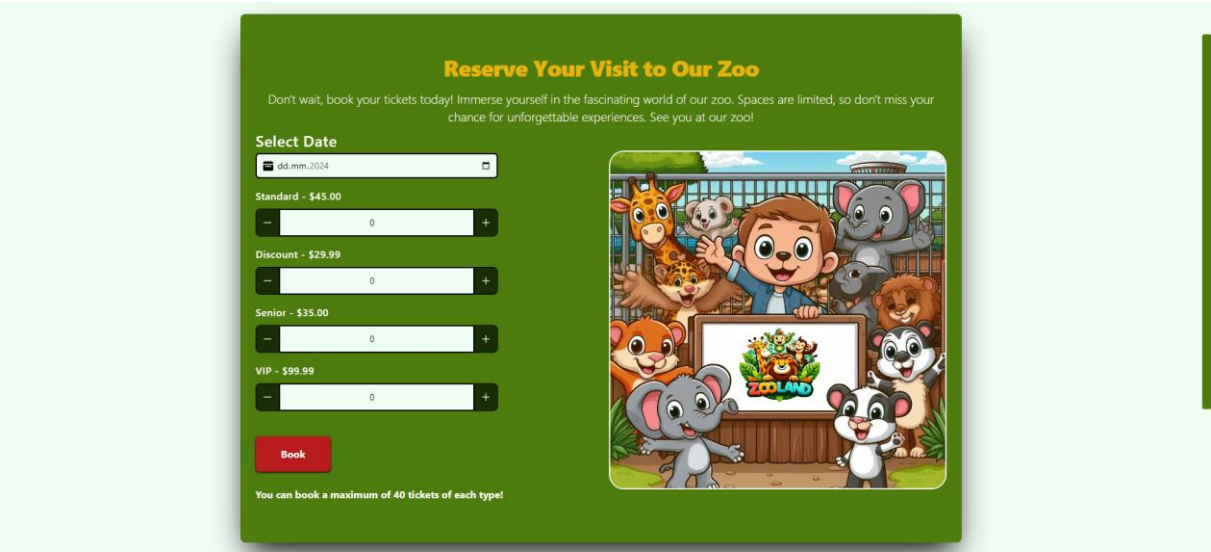
Senior: 3 ticket(s)

Book now

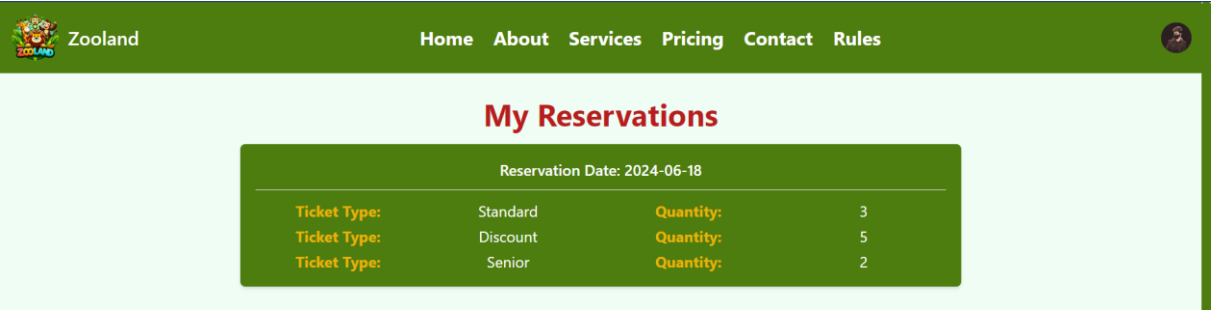
Already have an account? [Sign up](#)

Rysunek 7

Rezerwacje grupowe użytkownika:



Rysunek 8 - rezerwacja biletów



Rysunek 9 - zarezerwowane terminy

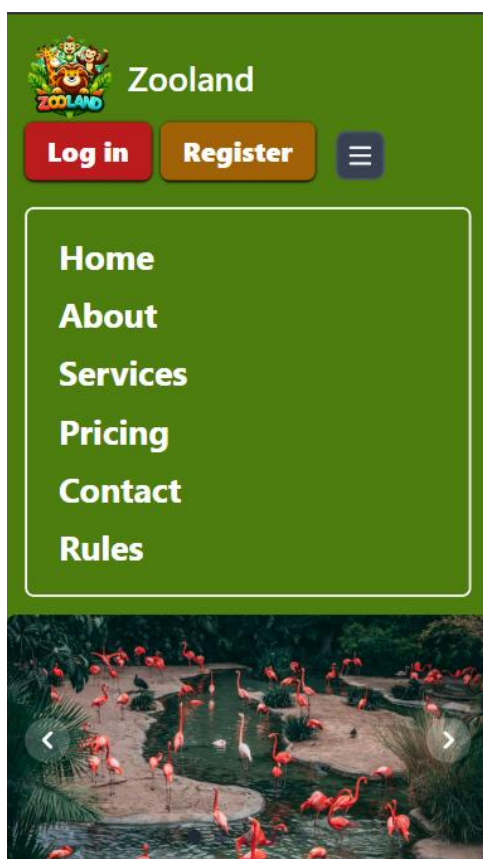
Mobilne

Dla prezentacji i przetestowania widoku mobilnego, wybrano urządzenie Iphone 14 Pro Max

Strona główna:



Rysunek 10 - panel nawigacyjny z karuzelą zdjęć





Rysunek 11 - rozwijane menu

Reserve Your Visit to Our Zoo

Don't wait, book your tickets today! Immerse yourself in the fascinating world of our zoo. Spaces are limited, so don't miss your chance for unforgettable experiences.

See you at our zoo!

Select Date

 dd.mm.2024 

Standard - \$45.00

—

0

+

Discount - \$29.99

—

0

+

Senior - \$35.00

—

0

+

VIP - \$99.99

—

0




+

Next

You can book a maximum of 5 tickets for each category. If you want to book a group visit, create an account!

Rysunek 12 - rezerwacja biletu



Rezerwacja grupowa:

**Zooland**



Reserve Your Visit to Our Zoo

Don't wait, book your tickets today! Immerse yourself in the fascinating world of our zoo. Spaces are limited, so don't miss your chance for unforgettable experiences. See you at our zoo!



Select Date

 dd.mm.2024



Standard - \$45.00



Discount - \$29.99

Senior - \$35.00


 

VIP - \$99.99




Book

You can book a maximum of 40 tickets of each type!



Rysunek 13

Zarządzanie zasobami użytkownika:

 **Zooland**  

Edit Profile

Name

Surname

Email


Password

Confirm Password

Avatar

Nie wybrano pliku

Your current avatar:



Rysunek 14

Uruchomienie aplikacji

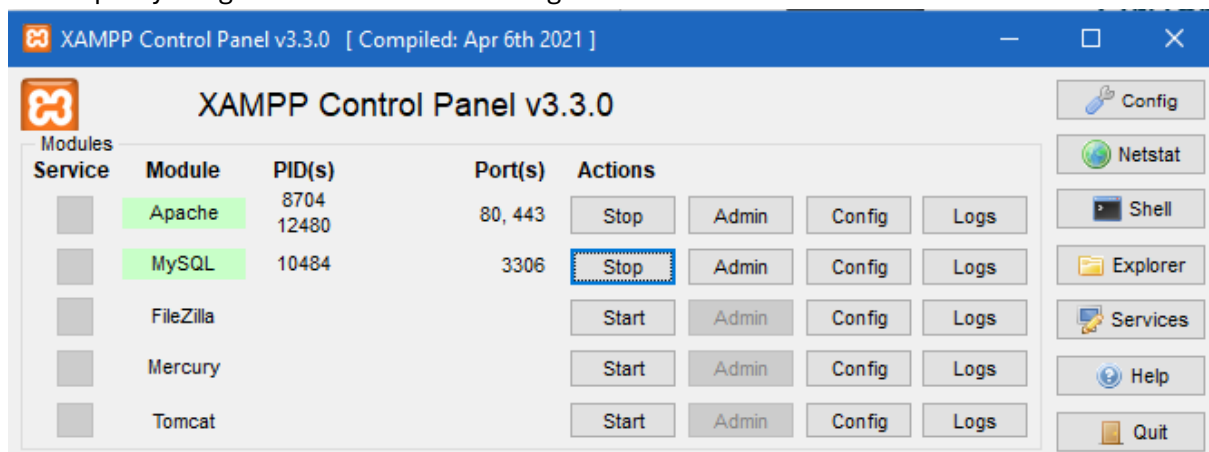
Aby mieć możliwość uruchomienia aplikacji należy mieć pobrane takie aplikacje jak:

- XAMPP
- Composer Setup
- Node.js
- Visual Studio Code

Należy podążać etapami instalacji aby pomyślnie uruchomić aplikację.

1. Pobrać aplikację z publicznego [repozytorium](#)
2. Rozpakować w wybranym przez siebie miejscu
3. Uruchomić XAMPP Control Panel i wystartować usługę serwer Apache oraz MySQL. Warto tutaj zwrócić uwagę na numer portu serwisu MySQL

Widok pomyślnego uruchomienia obu usług



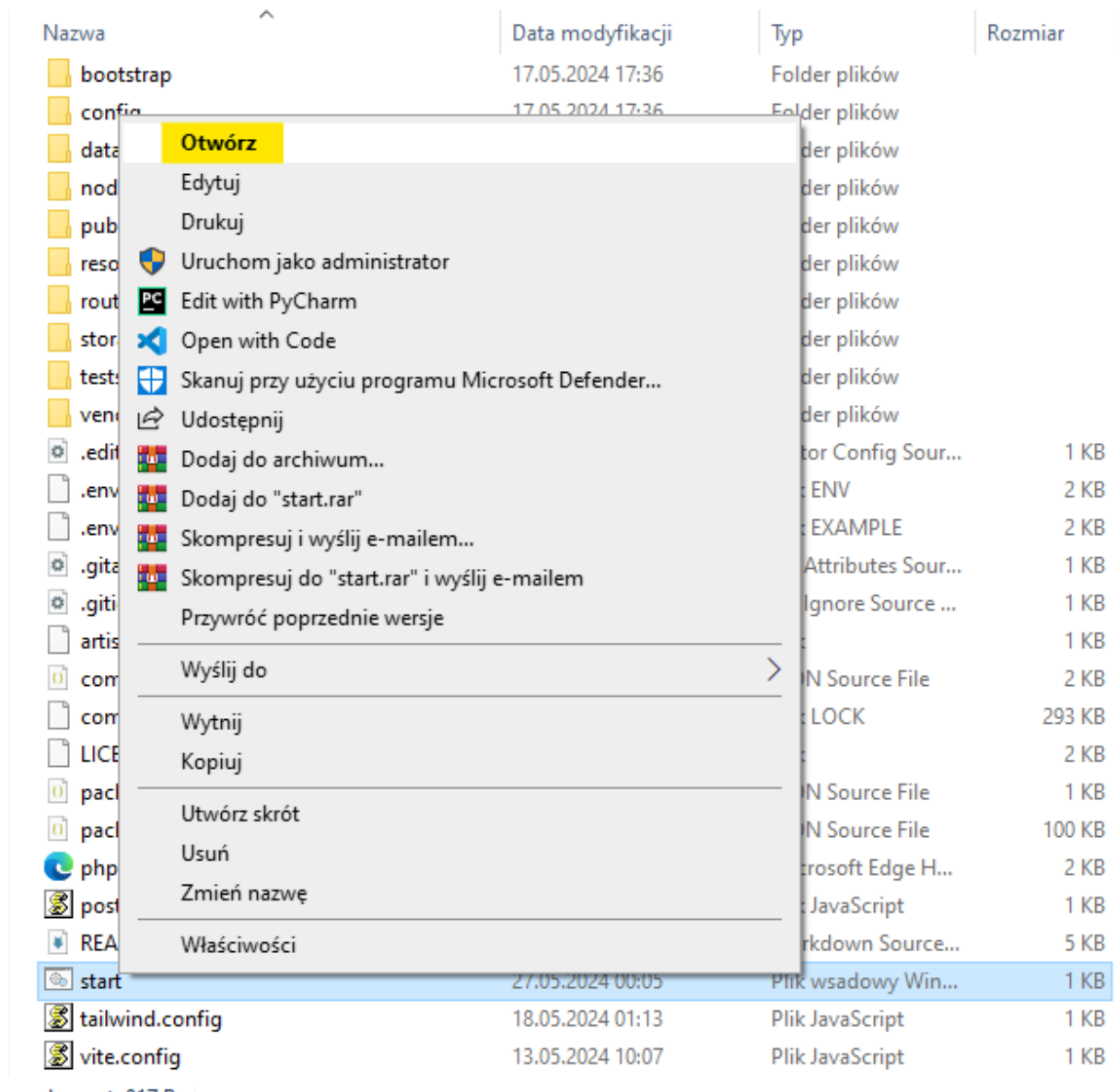
4. Ew. Dostosować port w pliku .env.example w miejscu DB_PORT = 3306, jeśli nie zgadza się on z portem MySQL w programie XAMPP

```
.env.example x
.env.example
22 DB_CONNECTION=mysql
23 DB_HOST=127.0.0.1
24 DB_PORT=3306
25 DB_DATABASE=zooland_db
26 DB_USERNAME=root
27 DB_PASSWORD=
28
```

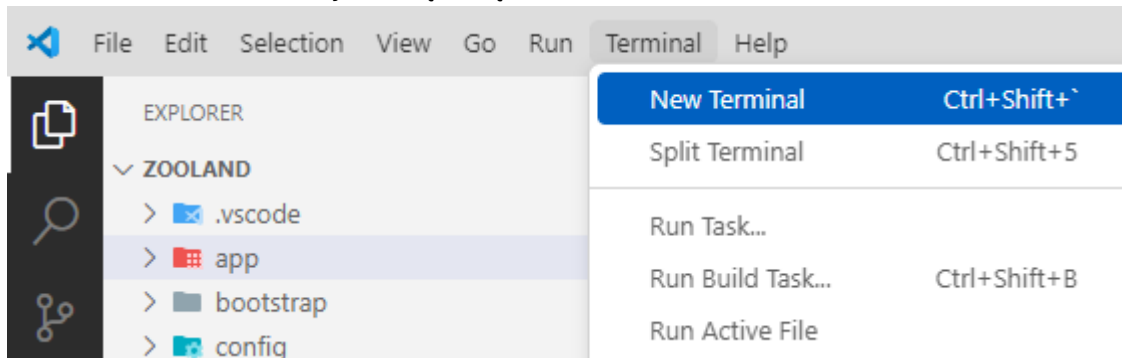
5. Uruchomić plik start.bat, który wykona:

- Utworzenie bazy danych, migracje oraz doda przykładowe seedery
- Utworzenie folderu vendor
- Instalacja Tailwind, zależności NPM oraz ApexCharts

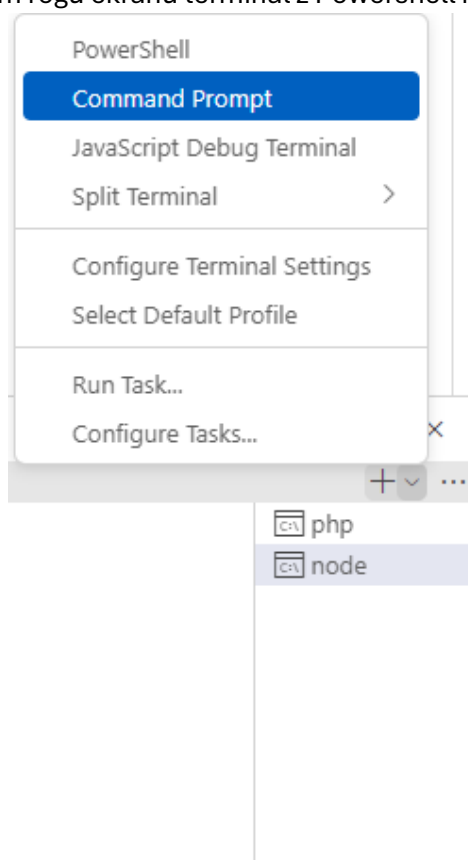
W przypadku ostrzeżeń, bądź błędów antywirusowych, należy je zignorować. Aplikacja nie zawiera złośliwego oprogramowania, ingeruje tylko w plikach programu XAMPP.



6. Otworzyć aplikację, pliki projektowe w środowisku Visual Studio Code. Przejść do zakładki Terminal i otworzyć nową kartę terminala



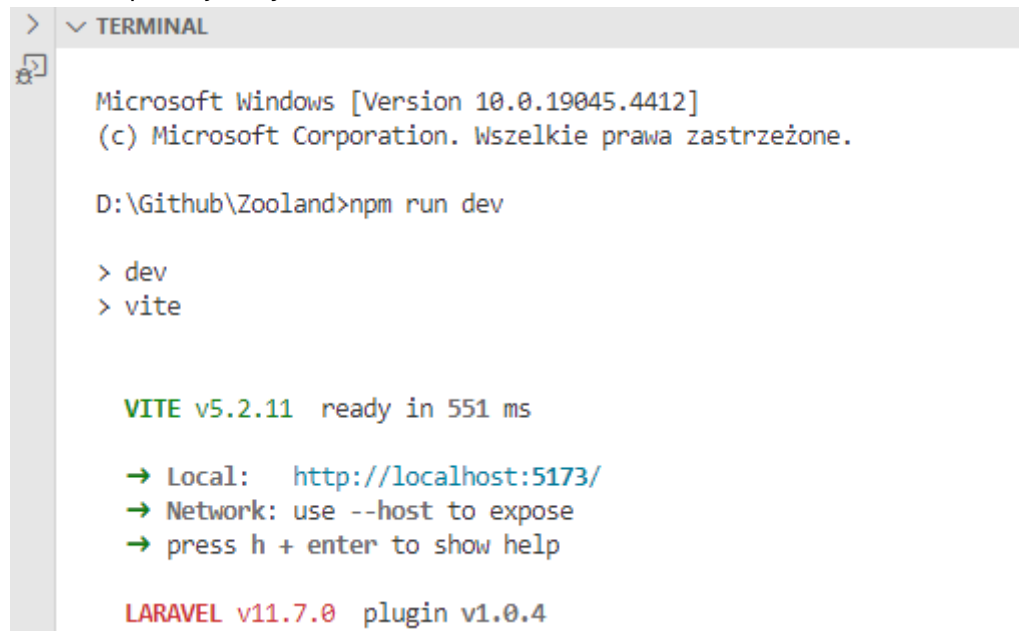
Zmienić w prawym dolnym rogu ekranu terminal z Powershell na Command Prompt



Wpisać polecenie `php artisan serve`, polecenie to uruchomi nam serwer.



7. Wpisać polecenie `npm run dev`, jest ono odpowiedzialne za skrypty kompilujące środowisko aplikacji, w tym Tailwind'a



```
> ▼ TERMINAL
Microsoft Windows [Version 10.0.19045.4412]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

D:\Github\Zooland>npm run dev

> dev
> vite

VITE v5.2.11  ready in 551 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help

LARAVEL v11.7.0  plugin v1.0.4
```

8. Otwórz przeglądarkę i przejdź pod adres <http://127.0.0.1:8000/>

Funkcjonalności aplikacji

Gość:

- Utworzenie konta
- Logowanie
- Przeglądanie ogólnodostępnych zasobów
- Rezerwacja biletów (ograniczona ilościowo do 5 sztuk każdego rodzaju biletu)
- Formularz kontaktowy z obsługą zoo

Użytkownik:

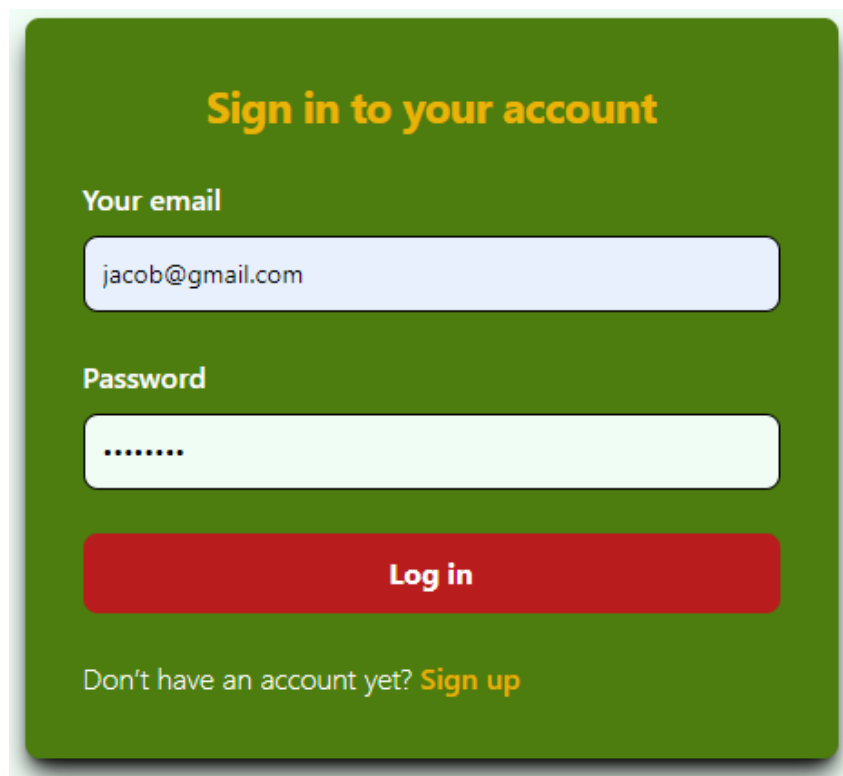
- Złożenie rezerwacji grupowych
- Wyświetlanie, usuwanie swoich rezerwacji
- Zarządzanie danymi
- Zarządzanie zdjęciem profilowym
- Formularz kontaktowy z obsługą zoo

Administrator:

- Edycja, usuwanie i dodawanie użytkowników
- Edycja, usuwanie i dodawanie typów/rodzajów biletów
- Edycja, usuwanie i dodawanie rezerwacji
- Edycja, usuwanie i dodawanie biletów dla konkretnych rezerwacji
- Zapis i wyświetlanie statystyk dla zarezerwowanych biletów

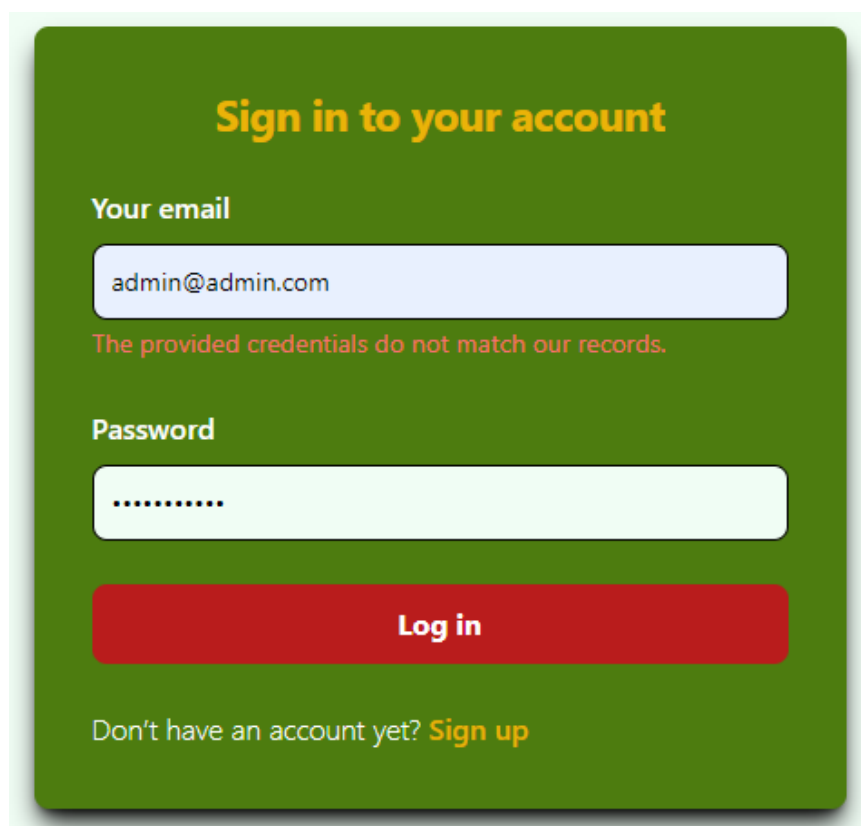
Logowanie

Etap logowania na użytkownika o loginie jacob@gmail.com oraz hasło – password, lub na konto administratora – admin@admin.com, hasło admin.



The image shows a login form with a green background. At the top, it says "Sign in to your account" in yellow. Below that, there are two input fields: "Your email" with the value "jacob@gmail.com" and "Password" with masked characters ".....". A red "Log in" button is below the password field. At the bottom, it says "Don't have an account yet? Sign up" in yellow.

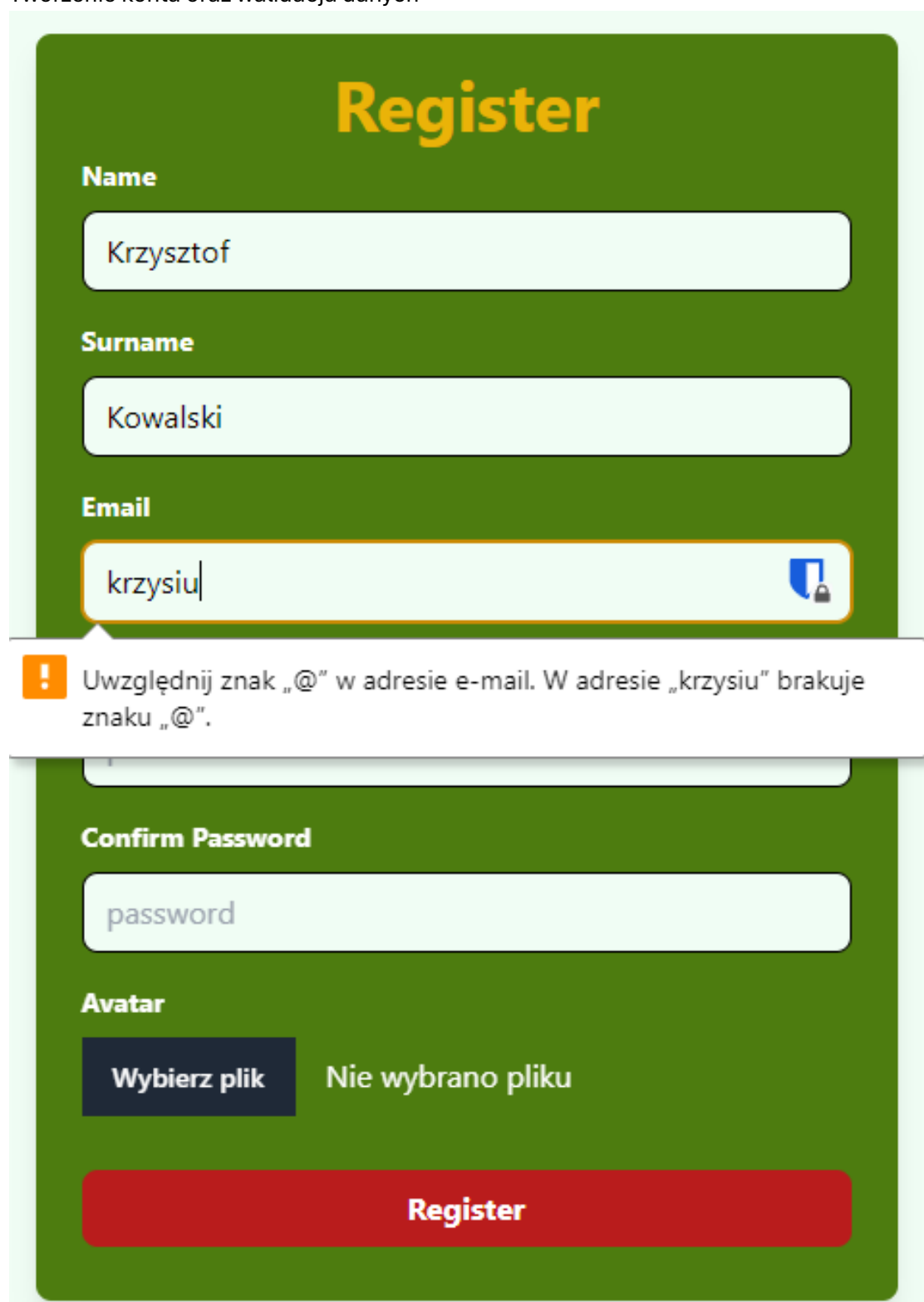
Komunikat w której użytkownik wprowadzi złe dane



The image shows the same login form as before, but with an error message. The "Your email" field now contains "admin@admin.com". Below the email field, there is a red error message: "The provided credentials do not match our records." The "Password" field still contains masked characters ".....". The "Log in" button and the "Sign up" link remain the same.

Rejestracja

Tworzenie konta oraz walidacja danych



The image shows a registration form titled "Register" on a green background. The form has several input fields: "Name" (containing "Krzysztof"), "Surname" (containing "Kowalski"), "Email" (containing "krzysiu"), "Confirm Password" (containing "password"), and "Avatar" (with a "Wybierz plik" button and the text "Nie wybrano pliku"). A red "Register" button is at the bottom. A validation error message is displayed in a white box with a red border and a red exclamation mark icon. The message reads: "Uwzględnij znak „@” w adresie e-mail. W adresie „krzysiu” brakuje znaku „@”." (Consider the "@" symbol in the email address. The "@" symbol is missing in the address "krzysiu").

Register

Name

Krzysztof

Surname

Kowalski

Email

krzysiu

! Uwzględnij znak „@” w adresie e-mail. W adresie „krzysiu” brakuje znaku „@”.

Confirm Password

password

Avatar

Wybierz plik Nie wybrano pliku

Register

Rysunek 15 - walidacja po stronie Front-Endu

Register

Name

Krzysztof

Surname

Kowalski

Email

krzysiu@gmail.com

Password

password

Confirm Password

password

The password field must be at least 5 characters.

Avatar

Wybierz plik

Nie wybrano pliku

Register

Rysunek 16 - walidacja po stronie Back-Endu

Funkcja zajmująca się rejestracją nowego użytkownika

```
22     public function showRegistrationForm()
23     {
24         return view('reservation.register');
25     }
26
33     public function register(Request $request)
34     {
35         $validator = Validator::make($request->all(), [
36             'name' => 'required|string|max:255',
37             'surname' => 'required|string|max:255',
38             'email' => 'nullable|string|email|max:255|unique:users',
39             'password' => 'required|string|min:5|confirmed',
40             'avatar' => 'nullable|image|mimes:jpeg,png,jpg|max:2048',
41         ]);
42
43         if ($validator->fails()) {
44             return redirect()->back()->withErrors($validator)->withInput();
45         }
46
47         //File upload
48         if ($request->hasFile('avatar'))
49         {
50             $avatarName = Str::random(20) . '.' . $request->file('avatar')->getClientOriginalExtension();
51             $request->file('avatar')->move(public_path('img/avatars'), $avatarName);
52         }
53         else
54         {
55             $avatarName = null;
56         }
57
58         // Check if email already exists
59         $existingUser = User::where('email', $request->input('email'))->first();
60         if ($existingUser) {
61             return redirect()->back()->withErrors(['email' => 'This email is already taken.'])->withInput();
62         }
63
64         // Store the user in the database
65         $user = User::create([
66             'name' => $request->input('name'),
67             'surname' => $request->input('surname'),
68             'email' => $request->input('email'),
69             'password' => Hash::make($request->input('password')),
70             'avatar' => $avatarName,
71             'role' => 'user',
72         ]);
73
74         // Log in the user
75         Auth::login($user);
76
77         return redirect()->route('home')->with('success', 'Registration successful! You are now logged in.');
```


Metoda `showRegistrationForm()` to metoda GET, która odpowiada za wyświetlanie widoku z formularzem rejestracji nowego użytkownika, zwracająca widok `'reservation.register'`

Metoda `register()` to metoda POST, która obsługuje dane przesłane z formularza rejestracji. Przyjmuje ona obiekt `Request` jako argument, która zawiera wszystkie dane przesłane z formularza. Metoda ta waliduje dane wejściowe, sprawdzając wszystkie wymagane pola takie jak `name`, `surname`, `email` czy jest unikalny, hasło które jest następnie wysyłane zaszyfrowane (funkcja `Hash::make()`) do bazy danych oraz avatar czy ma odpowiednie rozszerzenie. Walidacja wykonywana jest jako pierwsza przed wykonaniem następnych operacji na danych.

Avatar dodany przez użytkownika zapisywany jest w plikach źródłowych aplikacji w postaci losowych 20 znaków z odpowiednim rozszerzeniem.

Jeśli email jest już zajęty użytkownik otrzyma odpowiedni komunikat na stronie.

Tworzony jest nowy użytkownik za pomocą metody `User::create()`. Gdy zostanie dodany pomyślnie użytkownik zostanie przekierowany do strony głównej, w przeciwnym wypadku wraca na stronę rejestracji z odpowiednimi komunikatami.

Rezerwacja biletów przez gościa



Tworzenie nowej rezerwacji oraz walidacja

Reserve Your Visit to Our Zoo

Don't wait, book your tickets today! Immerse yourself in the fascinating world of our zoo. Spaces are limited, so don't miss your chance for unforgettable experiences.
See you at our zoo!

- The visit-date field is required.

Select Date

 dd.mm.2024 

Standard - \$45.00

-

3

+

Discount - \$29.99

-

2

+

Senior - \$35.00

-

5

+

VIP - \$99.99

-

0

+

Next

You can book a maximum of 5 tickets for each category. If you want to book a group visit, create an account!

Rysunek 17 - użytkownik nie wybrał daty rezerwacji

str. 24

Funkcja zajmująca się obsługą rezerwacji biletów gościa

```
11 class CalendarController extends Controller
12 {
13     public function reserve(Request $request)
14     {
15         $validator = Validator::make($request->all(), [
16             'visit-date' => [
17                 'required',
18                 function ($attribute, $value, $fail) {
19                     $invalidDates = ['01-01', '11-01', '12-24', '12-25'];
20
21                     // Dates prohibited check
22                     $selectedDate = date('m-d', strtotime($value));
23                     if (in_array($selectedDate, $invalidDates)) {
24                         $fail('The selected date is invalid!');
25                     }
26                 },
27             ],
28             'quantity' => 'required|array',
29             'quantity.*' => 'required|integer|min:0',
30         ]);
31
32         if ($validator->fails()) {
33             return redirect()->route('home', '#section3')->withErrors($validator)->withInput();
34         }
35
36         $visitDate = $request->input('visit-date');
37         $quantities = $request->input('quantity');
38         $allZero = true;
39         foreach ($quantities as $quantity)
40         {
41             if ($quantity > 0)
42             {
43                 $allZero = false;
44                 break;
45             }
46         }
47         if ($allZero) {
48             return redirect()->route('home', '#section3')->withErrors(['quantity' => 'You need to choose a ticket']);
49         }
50         $tickets = Ticket::all();
51
52         return view('reservation.submit', [
53             'visitDate' => $visitDate,
54             'quantities' => $quantities,
55             'tickets' => $tickets
56         ]);
57     }
58 }
59
```

Metoda reserve() jest metodą POST, pobiera dane wprowadzone w formularzu rezerwacji, sprawdza ponownie datę pod kątem dat w których zoo jest zamknięte np. święta Bożego Narodzenia. Pierwszy stopień walidacji odbywa się w widoku 'main.section' ([Rysunek 19 - walidacja kalendarza](#)), gdzie początkowa data to dzień następny od aktualnej daty a końcowa to zakres do 3 miesięcy.

Sprawdzana jest również ilość dla poszczególnego typu biletu, kiedy żaden bilet nie został wybrany zostanie zwrócony odpowiedni komunikat błędu dla gościa.

Gdy rezerwacja przebiegnie pomyślnie zwrócony zostanie widok 'reservation.submit', z przekazanymi do niej informacjami o rezerwacji.

Przykładowy widok potwierdzenia rezerwacji

The image shows a reservation confirmation form with a green background. The form contains the following fields and elements:

- First name:** Celina
- Last name:** Nowak
- Email address:** celina.nowak@gmail.com
- Terms and conditions:** A checkbox is unchecked, and a tooltip message says: "Zaznacz to pole, jeśli chcesz kontynuować." (Check this field if you want to continue).
- Event Date:** 2024-06-21
- Ticket summary:**
 - Standard: 4 ticket(s)
 - Discount: 2 ticket(s)
 - Senior: 2 ticket(s)
- Buttons:** "Book now" (red) and "Sign up" (yellow).
- Footer:** "Already have an account? Sign up"

Rysunek 20 - walidacja danych dla rezerwacji gościa

Dane do rezerwacji biletów przez gościa dodawane są do bazy wykorzystując już wcześniej wspomniane pola – guest_name, guest_surname, guest_email.

W ten sposób administratorzy czy obsługa zoo są poinformowani o faktycznym stanie rezerwacji i czy należy dokonać ponownych sprawdzenia danych w punktach obsługi zoo.

Funkcja finalna odpowiadająca za rezerwację z danymi gościa

```
10 class GuestReservationController extends Controller
11 {
12     public function submit(Request $request)
13     {
14         $validator = Validator::make($request->all(), [
15             'reservation_date' => 'required|date',
16             'quantity' => 'required|array',
17             'quantity.*' => 'required|integer|min:0',
18             'guest_name' => 'required|string|max:255',
19             'guest_surname' => 'required|string|max:255',
20             'guest_email' => 'required|email|max:255',
21         ]);
22
23         if ($validator->fails()) {
24             return redirect()->route('home', '#section3')->withErrors($validator)->withInput();
25         }
26
27         $visitDate = $request->input('reservation_date');
28         $quantities = $request->input('quantity');
29         $firstName = $request->input('guest_name');
30         $lastName = $request->input('guest_surname');
31         $email = $request->input('guest_email');
32
33         $reservation = new Reservation();
34         $reservation->guest_name = $firstName;
35         $reservation->guest_surname = $lastName;
36         $reservation->guest_email = $email;
37         $reservation->reservation_date = $visitDate;
38         $reservation->save();
39
40         foreach ($quantities as $ticketId => $quantity) {
41             if ($quantity > 0) {
42                 TicketUser::create([
43                     'reservation_id' => $reservation->id,
44                     'ticket_id' => $ticketId,
45                     'quantity' => $quantity,
46                 ]);
47             }
48         }
49
50         return redirect()->route('home', '#section3')->with('status', 'Reservation successfully made!');
51     }
52 }
```

Metoda submit() to metoda POST, która sprawdza ponownie datę rezerwacji oraz ilość biletów danego typu. Walidacja również jest wykonana na wprowadzonych danych personalnych gościa. Jeśli dane są prawidłowe to rezerwacja zostaje dodana do bazy danych, w innym przypadku zwraca błąd na stronie.

New Reservation() tworzy nową rezerwację uzupełnioną danymi wprowadzonymi przez gościa, następnie za pomocą metody save() dodawana jest do bazy danych.

TicketUser::create() to metoda która wykonywana jest w pętli dla dokonanej rezerwacji. Dodaje ona poszczególne bilety do bazy danych, wiążąc zarezerwowane bilety dla ogółu rezerwacji.

Rezerwacja wizyt grupowych

Widok oraz walidacja rezerwacji dokonanej przez zalogowanego użytkownika

Reserve Your Visit to Our Zoo

Don't wait, book your tickets today! Immerse yourself in the fascinating world of our zoo. Spaces are limited, so don't miss your chance for unforgettable experiences. See you at our zoo!

You need to choose a ticket

Select Date

20.06.2024

Standard - \$45.00

—

0

+

Discount - \$29.99

—

0

+

Senior - \$35.00

—

0

+

VIP - \$99.99


—

0

+

Book

You can book a maximum of 40 tickets of each type!



Rysunek 21- walidacja, użytkownik nie wprowadził ilości biletów

Kalendarz wykorzystywany w tym miejscu ma tę samą obsługę walidacji, która wykonuje się po stronie klienta co kalendarz gościa ([Rysunek 19 - walidacja kalendarza](#)).

Funkcja odpowiedzialna za dodawanie rezerwacji grupowej

```
32 public function reserve(Request $request)
33 {
34     $quantities = $request->input('quantity');
35     $allZero = true;
36     foreach ($quantities as $quantity)
37     {
38         if ($quantity > 0)
39         {
40             $allZero = false;
41             break;
42         }
43     }
44     if ($allZero) {
45         return redirect()->back()->withErrors(['quantity' => 'You need to choose a ticket']);
46     }
47
48     $request->validate([
49         'visit-date' => 'required|date',
50         'quantity' => 'required|array',
51         'quantity.*' => 'integer|min:0',
52     ]);
53
54     $reservation = new Reservation();
55     $reservation->user_id = Auth::id();
56     $reservation->reservation_date = $request->input('visit-date');
57     $reservation->save();
58
59     foreach ($request->input('quantity') as $ticketId => $quantity) {
60         if ($quantity > 0) {
61             DB::table('ticket_user')->insert([
62                 'reservation_id' => $reservation->id,
63                 'ticket_id' => $ticketId,
64                 'quantity' => $quantity,
65                 'created_at' => now(),
66                 'updated_at' => now()
67             ]);
68         }
69     }
70
71     return redirect()->route('user-reservation')->with('status', 'Reservation successful!');
72 }
```

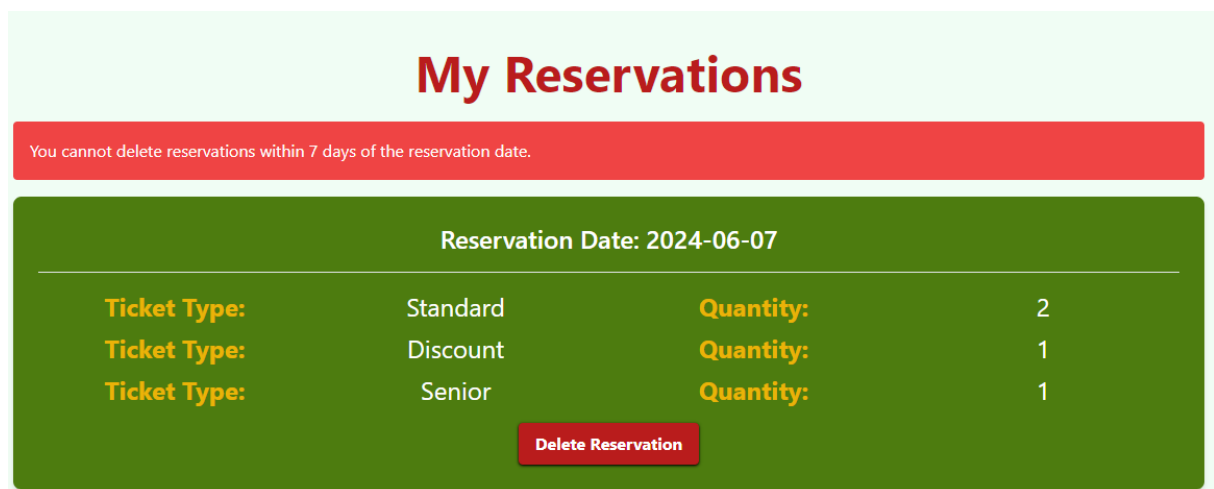
Metoda reserve() to metoda POST, która sprawdza czy użytkownik wprowadził ilość dla biletów, jeśli nie to zwraca widok z komunikatem. Wykonuje ona również walidację daty i ilości

New Reservation() tworzy nową rezerwację uzupełnioną danymi wprowadzonymi przez użytkownika i przypisuje mu ją, następnie za pomocą metody save() dodawana jest do bazy danych.

Dodawanie biletów do tablicy 'ticket_user' wykonywane jest w pętli aby przypisać każdy typ biletu i jego ilość dla odpowiedniej, wykonanej przez użytkownika rezerwacji wizyty grupowej.

Zarządzanie rezerwacjami użytkownika

Użytkownik może wyświetlać swoje rezerwacje a także nimi zarządzać, o ile spełnione są odpowiednie reguły.



Rysunek 22 - próba usunięcia rezerwacji, która ma mniejszy termin niż 7 dni

Odpowiedzialna jest za to metoda `destroy()` typu DELETE

```
74 public function destroy($id)
75 {
76     $reservation = Reservation::find($id);
77     if (!$reservation || $reservation->user_id != Auth::id()) {
78         return redirect()->route('user.reservations')->withErrors('Reservation not found or you do not have permission to delete it.');
```

Metoda `Reservation::find($id)`, sprawdza czy istnieje taka rezerwacja którą użytkownik zamierza usunąć, jeśli nie zwróci odpowiedni komunikat.

Użytkownik nie może usunąć następujących rezerwacji:

- Rezerwacja z przeszłości, której termin już minął
- Rezerwacja do której zostało mniej niż 7 dni (włącznie)
- Rezerwacja której numer id nie zgadza się z tym z bazy danych

Jeśli rezerwacja spełnia warunki to metoda `delete()` usuwa rezerwację i powiązane z nią bilety w bazie danych oraz zwraca widok z informacją o sukcesie.

Zasoby zalogowanego użytkownika

Każdy użytkownik po zalogowaniu może zarządzać swoimi zasobami, danymi. Po kliknięciu w swoje zdjęcie profilowe -> Setting.

Edit Profile

Name

Surname

Email

Password

Confirm Password


The password field confirmation does not match.

Avatar

Wybierz plik

Nie wybrano pliku

Your current avatar:



Update

Rysunek 23 - przykład walidacji dla hasła

Funkcja odpowiedzialna za aktualizację danych użytkownika

```
25 public function update(Request $request)
26 {
27     $user = User::find(Auth::id());
28
29     $validator = Validator::make($request->all(), [
30         'name' => 'required|string|max:255',
31         'surname' => 'required|string|max:255',
32         'email' => 'required|string|email|max:255|unique:users,email,' . $user->id,
33         'password' => 'nullable|string|min:5|confirmed',
34         'avatar' => 'nullable|image|mimes:jpeg,png,jpg|max:2048',
35     ]);
36
37     if ($validator->fails()) {
38         return redirect()->back()->withErrors($validator)->withInput();
39     }
40
41     $userData = [
42         'name' => $request->input('name'),
43         'surname' => $request->input('surname'),
44         'email' => $request->input('email'),
45     ];
46
47     if ($request->filled('password')) {
48         $userData['password'] = Hash::make($request->input('password'));
49     }
50 }
```

Metoda update() to metoda typu POST. Wyszukiwany jest użytkownik o przekazanym id aby wypełnić formularz danymi. Pola password są nie wypełnione, użytkownik wypełnia je wtedy kiedy chce zmienić hasło na nowe.

Walidacja wykonywana jest kiedy wprowadzone są nowe lub zaktualizowane dane. Sprawdzane jest:

- Name, surname – wymagane, pole typu tekstowego, maksymalna ilość znaków 255
- Email – wymagany, pole typu tekstowego, maksymalna ilość znaków 255, pole unikalne
- Password – może być puste, pole typu tekstowego, minimalna ilość znaków 5, wymagane potwierdzenie
- Avatar – plik typu graficznego, rozszerzenie: jpeg,png,jpg, maksymalna waga 2MB

Zaktualizowane hasło zostaje zaszyfrowane za pomocą funkcji Hash::make(), wtedy i tylko wtedy gdy użytkownik wprowadzi nowe hasło w obu polach oraz spełni wymóg walidatora.

Dla zarządzania avatarami użytkownika wykorzystano następujące metody

```
51     if($request->hasFile('avatar'))
52     {
53         if($user->avatar)
54         {
55             $avatarPath = public_path('img/avatars/') . $user->avatar;
56             if(file_exists($avatarPath))
57             {
58                 unlink($avatarPath);
59             }
60         }
61     }
62
63     if ($request->hasFile('avatar')) {
64         $avatarName = Str::random(20) . '.' . $request->file('avatar')->getClientOriginalExtension();
65         $request->file('avatar')->move(public_path('img/avatars'), $avatarName);
66         $userData['avatar'] = $avatarName;
67     }
68
69     $user->update($userData);
70
71     return redirect()->route('settings')->with('success', 'Profile updated successfully!');
72 }
--
```


Sprawdzić czy użytkownik przekazał nowe zdjęcie profilowe, jeśli tak to usuwane jest poprzednie z plików projektowych, żeby zastąpić je nowym.

Dodanie nowego avatara, przesłanego przez użytkownika do plików aplikacji. Nadanie mu nazwy poprzez losowanie 20 znaków oraz dodanie pierwotnego rozszerzenia. W bazie danych przetrzymywana będzie sama nazwa zdjęcia profilowego użytkownika w postaci 'nazwa.jpg'.

Metoda update() wykonuje aktualizację danych użytkownika w bazie danych po pomyślnym ukończeniu walidacji i operacji wykonanych na awatarze. Zwrócenie widoku z komunikatem o sukcesie aktualizacji profilu.

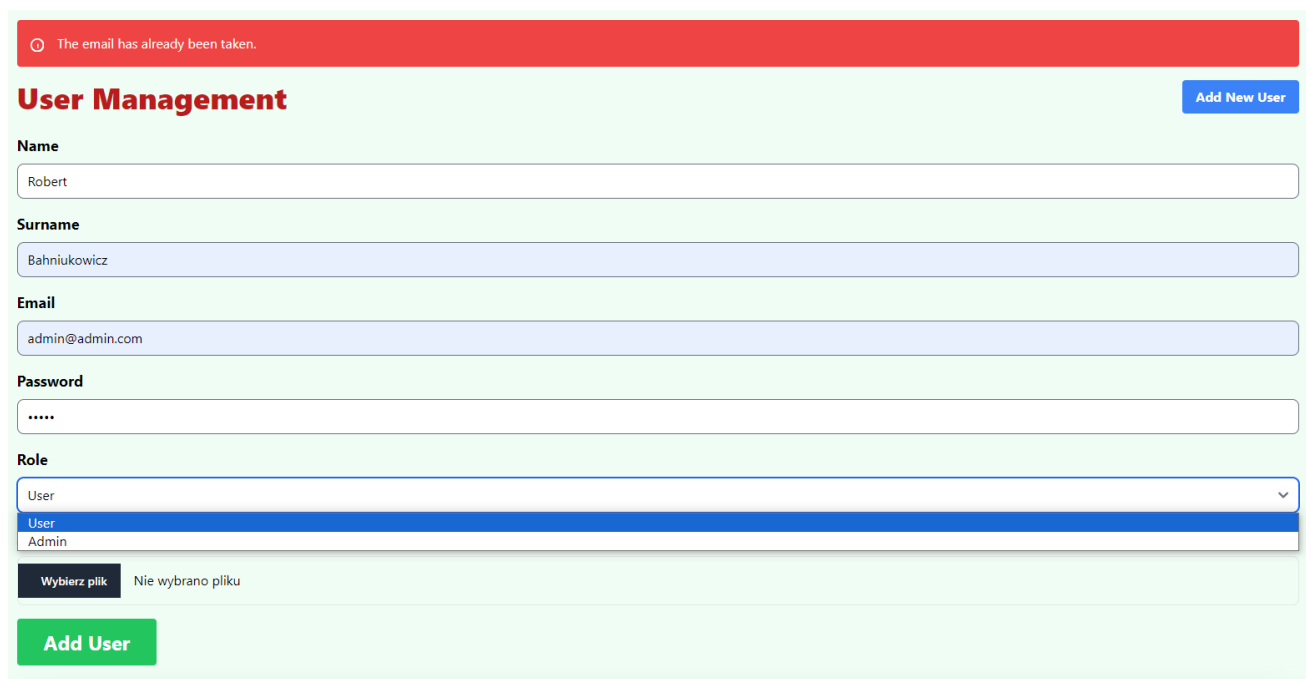
Zarządzanie użytkownikami

Administrator zarządza w dowolny sposób użytkownikami, ma możliwość edycji danych, usunięcia bądź dodania użytkowników.



The screenshot shows a 'User Management' interface. At the top left is the title 'User Management' in red. At the top right is a blue button labeled 'Add New User'. Below the title is a green card containing user details: 'Name: Jacob', 'Surname: Jacobowski', 'Role: user', and 'Email: jacob@gmail.com'. There is a circular profile picture of a person with a beard. At the bottom right of the green card are two buttons: 'Edit' (yellow) and 'Delete' (red).

Rysunek 24 - główny widok zarządzania użytkownikami



The screenshot shows the 'User Management' interface with a form to add a new user. At the top, there is a red error message: 'The email has already been taken.' Below the title 'User Management' is a blue button labeled 'Add New User'. The form fields are: 'Name' (text input with 'Robert'), 'Surname' (text input with 'Bahniukowicz'), 'Email' (text input with 'admin@admin.com'), 'Password' (password input with '.....'), and 'Role' (dropdown menu with 'User' selected). Below the role dropdown is a file upload area with a button 'Wybierz plik' and the text 'Nie wybrano pliku'. At the bottom is a green button labeled 'Add User'.

Rysunek 25 - próba dodania nowego użytkownika z wcześniej zajęтым emailem

Administrator ma dostęp do wyboru roli dla danego, bądź nowego użytkownika. Do wyboru są dwie role – User (domyślna) oraz Admin.

Administrator nie może edytować z widoku aplikacji danymi dla administratora głównego (id=1), w celach bezpieczeństwa. Aby edytować te dane należy to zrobić po stronie bazy danych, tylko i wyłącznie na własne ryzyko.

User Management


Add New User

Name: Jacob

Surname: Jacobowski

Role: user

Email: jacob@gmail.com



Edit

Delete

Name

Jacob

Surname

Jacobowski

Email

jacob@gmail.com

Password

Leave blank to keep current password

Role

User

Avatar

Wybierz plik

Nie wybrano pliku

Save

Cancel

Rysunek 26 - pełny widok zarządzania wybranym użytkownikiem

Administrator może zarządzać istniejącym użytkownikiem, pola automatycznie są generowane na podstawie jego danych. Pole na hasło zawsze jest puste, brak zamiaru zmiany hasła dla użytkownika to zostawienie tego pola pustego, w przypadku wpisania tam nowych danych, hasło zostanie zaszyfrowane i dodane do bazy.

Zmiana avatara jest opcjonalna, w przypadku kiedy użytkownik ma zdjęcie profilowe, które wyraźnie nie przestrzega regulaminu, ma kontekst niemoralny należy zmienić je na podstawowe które znajduje się w plikach z innymi avatarami pod nazwą 'default.png'

Funkcja odpowiedzialna za dodawanie nowych użytkowników, metodą typu POST

```
22 // Method for create new user
23 public function store(Request $request)
24 {
25     $request->validate([
26         'name' => 'required|string|max:255',
27         'surname' => 'required|string|max:255',
28         'email' => 'required|string|email|max:255|unique:users',
29         'password' => 'required|string|min:5',
30         'role' => 'required|string|in:user,admin',
31         'avatar' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:1024',
32     ]);
33
34     // check if email is unique
35     $validator = Validator::make($request->all(), [
36         'email' => 'unique:users,email'
37     ]);
38
39     // email isnt unique
40     if ($validator->fails()) {
41         return back()->withErrors(['email' => 'Email already exists.'])->withInput();
42     }
43
44     // create new user from inputs
45     $user = new User();
46     $user->name = $request->name;
47     $user->surname = $request->surname;
48     $user->email = $request->email;
49     $user->password = bcrypt($request->password);
50     $user->role = $request->role;
51
52     // add new avatar
53     if ($request->hasFile('avatar')) {
54         $avatar = $request->file('avatar');
55         $avatarName = Str::random(20) . '.' . $avatar->getClientOriginalExtension();
56         $avatar->move(public_path('img/avatars'), $avatarName);
57         $user->avatar = $avatarName;
58     }
59
60     $user->save();
61
62     return redirect()->route('userCRUD')->with('success', 'User added successfully.');
```

Validator sprawdza te dane które zostały zaktualizowane, jeśli nie to używa aktualnych. Możliwość zmiany roli użytkownika, jeżeli zmieniona to jest pobierana z formularza.

Sprawdzanie czy zmieniony email jest unikalny tj. nikt z zarejestrowanych użytkowników w bazie nie ma takiego emaila. Jest to unikalny rodzaj danych dla każdego użytkownika, tak aby każdy miał dostęp do swojego konta względem struktury bazy danych (Rysunek 1 - diagram ERD).

Dodawanie nowego avatara, jeśli został wprowadzony nowy i usunięcie starego w plikach projektowych (public/img/avatars).

Funkcja save() zapisuje nam zaktualizowane dane użytkownika jeśli proces przebiegł pomyślnie, w innych przypadkach zwrócony zostaje odpowiedni komunikat błędu.

Funkcja destroy(), obsługująca usunięcie użytkownika

```
122     public function destroy(User $user)
123     {
124         try
125         {
126             $user->delete();
127             return redirect()->route('userCRUD')->with('success', 'User deleted successfully.');
```

Całość przetwarzana w bloku 'try-catch', spowodowane to jest tym iż, istnieje możliwość aby dany użytkownik miał istniejące powiązane rezerwacje ze sobą. W takich przypadkach odmowy przez istniejące rezerwacje użytkownika, najpierw należy usunąć jego rezerwacje a potem powrócić do pierwotnych planów.

Funkcja update() zawiera to samo co metoda store() z wyjątkiem zarządzania avatarem

```
99     if ($request->hasFile('avatar')) {
100         if ($user->avatar) {
101             $oldAvatarPath = public_path('img/avatars/' . $user->avatar);
102             if (file_exists($oldAvatarPath)) {
103                 unlink($oldAvatarPath);
104             }
105         }
106         $avatar = $request->file('avatar');
107         $avatarName = Str::random(20) . '.' . $avatar->getClientOriginalExtension();
108         $avatar->move(public_path('img/avatars'), $avatarName);
109
110         $userData['avatar'] = $avatarName;
111     }
112 }
```

Najpierw wykonywane jest usunięcie poprzedniego zdjęcia profilowego, przez zastąpieniem go nowym zdjęciem. Ponownie jak w innych przypadkach, aktualizacja avatara wiąże się z zapisem go w plikach projektowych pod nazwą złożoną z 20 losowych znaków + rozszerzenie pliku.

Zarządzanie rodzajami biletów

Administrator ma możliwość zarządzania rodzajami/typami biletów które następnie są rezerwowane przez użytkowników.

Ticket Management

[Add New Ticket](#)

Type: Standard

Price: 45.00

Available Quantity: 200

EditDelete

Type: Discount

Price: 29.99

Available Quantity: 100

EditDelete

Type: Senior

Price: 35.00

Available Quantity: 100

EditDelete

Type: VIP

Price: 99.99

Available Quantity: 10

EditDelete

Rysunek 27 - podstawowy widok prezentujący zarządzanie dostępnymi typami biletów

Ticket Management

Add New Ticket

Type

Price

Available Quantity

Add Ticket

Rysunek 28 - dodawanie nowego biletu

Type: Senior

Price: 35.00

Available Quantity: 100

Edit

Delete

Type

Senior

Price

35.00

Available Quantity

100

Save

Cancel

Rysunek 29 - edycja istniejącego biletu

Komunikaty walidacji:

Ticket created successfully.

Ticket Management

Add New Ticket

Rysunek 30 - pomyślnie dodany bilet

Ticket deleted successfully.

Ticket Management

Add New Ticket

Rysunek 31- pomyślne usunięty bilet

Funkcje do wyświetlania i dodawania nowych biletów

```
11 public function index()
12 {
13     $tickets = Ticket::all();
14     return view('admin.ticketCRUD', compact('tickets'));
15 }
16
17 public function store(Request $request)
18 {
19     $request->validate([
20         'type' => 'required',
21         'price' => 'required|numeric|min:0',
22         'available_quantity' => 'required|integer|min:0',
23     ]);
24
25     Ticket::create($request->all());
26
27     return redirect()->route('admin.tickets')->with('success', 'Ticket created successfully.');
```

Metoda index() typu GET, zwraca nam widok z biletami pobranymi z bazy danych.

Metoda store() typu POST obsługuje nam walidacje pól takich jak typ biletu, cena oraz dostępna ilość. Za pomocą Ticket::create() tworzony jest nowy bilet oraz zostaje dodany do bazy danych. Jeśli operacja przeszła pomyślnie, zwracany jest widok z komunikatem o pomyślnym dodaniu.

```
30 public function update(Request $request, Ticket $ticket)
31 {
32     $request->validate([
33         'type' => 'required',
34         'price' => 'required|numeric|min:0',
35         'available_quantity' => 'required|integer|min:0',
36     ]);
37
38     $ticket->update($request->all());
39
40     return redirect()->route('admin.tickets')->with('success', 'Ticket updated successfully.');
```

```
41 }
42
43 public function destroy(Ticket $ticket)
44 {
45     try
46     {
47         $ticket->delete();
48         return redirect()->route('admin.tickets')->with('success', 'Ticket deleted successfully.');
```

```
49     }
50     catch(\Illuminate\Database\QueryException $e)
51     {
52         if($e->getCode() === '23000')
53         {
54             return redirect()->route('admin.tickets')->with('error', 'You cannot delete a ticket associated with an existing registration');
```

```
55         }
56         else
57         {
58             return redirect()->route('admin.tickets')->with('error', 'An unexpected error occurred.');
```

```
59         }
60     }
61 }
62
63 }
```

Metoda update() typu POST aktualizuje zmienione dane dla danego biletu. Po prawidłowej walidacji aktualizuje dany bilet w bazie danych oraz zwraca widok z komunikatem o sukcesie.

Metoda destroy() typu DELETE usuwa bilet z bazy danych. W kontekście wykorzystania biletu w istniejącej rezerwacji, dodano obsługę bezpieczeństwa usuwania. Kiedy bilet jest już wykorzystywany w rezerwacji, strona zwraca widok z komunikatem błędu.

Zarządzanie rezerwacjami

Administrator ma pełny dostęp do zarządzania rezerwacjami. Edycja, usuwanie czy dodawanie rezerwacji jest obsługiwane przez widok który pozwala na wykonanie tych operacji po stronie aplikacji.

Reservation Management Add New Reservation

Reservation number: 2

Guest Name: Mateusz

Guest Surname: Kowalski

Guest Email: Mateusz@gmail.com

Reservation Date: 2024-05-31

Edit Delete

Rysunek 32 - przykładowy główny widok zarządzania rezerwacjami

Reservation number: 3

User Name: Jacob

User Surname: Jacobowski

User Email: jacob@gmail.com

Reservation Date: 2024-06-07

Edit Delete

User

Jacob Jacobowski (jacob@gmail.com) ▾

Guest Name

Guest Surname

Guest Email

Reservation Date

07.06.2024 ▾

Save Cancel

Rysunek 33 - edycja istniejącej rezerwacji

Reservation Management

Add New Reservation

User

-- Select User --

Guest Name

Guest Surname

Guest Email

Reservation Date

dd.mm.2024

Add Reservation

Rysunek 34 - dodawanie nowej rezerwacji

Komunikaty walidacji:

Reservation deleted successfully.

Reservation Management

Add New Reservation

Rysunek 35 - pomyślne usunięcie rezerwacji

Reservation added successfully.

Reservation Management

Add New Reservation

Rysunek 36 - pomyślne dodanie nowej rezerwacji

Obsługa dodawania nowej rezerwacji przez dwa typy użytkownika, zarejestrowanego (wybór z listy rozwijanej) lub dodanie rezerwacji dla gościa. Administrator wybierając użytkownika z listy, ma zablokowany formularz dla pól gościa. Obsługę tę wykonano za pomocą języka JavaScript

```
168     <script>
169         function toggleAddReservationForm() {
170             var form = document.getElementById('addReservationForm');
171             form.classList.toggle('hidden');
172         }
173
174         function toggleEditReservationForm(reservationId) {
175
176             var form = document.getElementById('editReservationForm_' + reservationId);
177             form.classList.toggle('hidden');
178         }
179
180         function handleUserSelection()
181         {
182             var userSelect = document.getElementById('user_id');
183             var selectedValue = userSelect.value;
184             var guestNameInput = document.getElementById('guest_name');
185             var guestSurnameInput = document.getElementById('guest_surname');
186             var guestEmailInput = document.getElementById('guest_email');
187
188             if (userSelect.value !== '') {
189                 guestNameInput.disabled = true;
190                 guestSurnameInput.disabled = true;
191                 guestEmailInput.disabled = true;
192                 guestNameInput.value = '';
193                 guestSurnameInput.value = '';
194                 guestEmailInput.value = '';
195                 guestNameInput.classList.add('cursor-not-allowed');
196                 guestSurnameInput.classList.add('cursor-not-allowed');
197                 guestEmailInput.classList.add('cursor-not-allowed');
198             } else {
199                 guestNameInput.disabled = false;
200                 guestSurnameInput.disabled = false;
201                 guestEmailInput.disabled = false;
202                 guestNameInput.classList.remove('cursor-not-allowed');
203                 guestSurnameInput.classList.remove('cursor-not-allowed');
204                 guestEmailInput.classList.remove('cursor-not-allowed');
205             }
206             console.log(selectedValue);
207         }
208     </script>
209
```

Funkcja toggleAddReservationForm() obsługuje dynamiczne wyświetlanie formularza dodawania nowej rezerwacji dla widoku.

Funkcja toggleEditReservationForm() obsługuje dynamiczne wyświetlanie formularza edycji istniejącej rezerwacji dla widoku.

Funkcja handleUserSelection() dynamicznie blokuje/odblokuje pola formularza dodawania oraz edycji w zależności od typu użytkownika : gość lub zarejestrowany użytkownik

Funkcje wyświetlania i dodawania rezerwacji

```
11 public function index()
12 {
13     $users = User::all();
14     $reservations = Reservation::with('user')->get();
15     return view('admin.reservationCRUD', compact('reservations', 'users'));
16 }
17
18 public function store(Request $request)
19 {
20     $request->validate([
21         'user_id' => 'nullable|exists:users,id',
22         'guest_name' => 'string|max:255',
23         'guest_surname' => 'string|max:255',
24         'guest_email' => 'email|max:255',
25         'reservation_date' => 'required|date',
26     ]);
27
28     Reservation::create($request->all());
29
30     return redirect()->route('admin.reservations')->with('success', 'Reservation added successfully.');
```

Metoda index() typu GET, pobiera dane wszystkich użytkowników oraz rezerwacji i zwraca je w widoku 'admin.reservationCRUD'.

Metoda store() typu POST służy do dodawania nowych rezerwacji. Wykorzystuje ona walidację gdzie sprawdza w zależności od typu: pola gościa albo id użytkownika oraz datę rezerwacji.

Funkcja aktualizacji i usuwania rezerwacji

```
34 public function update(Request $request, Reservation $reservation)
35 {
36     $request->validate([
37         'user_id' => 'nullable|exists:users,id',
38         'guest_name' => 'required|string|max:255',
39         'guest_surname' => 'required|string|max:255',
40         'guest_email' => 'required|email|max:255',
41         'reservation_date' => 'required|date',
42     ]);
43
44     $reservation->update($request->all());
45
46     return redirect()->route('admin.reservations')->with('success', 'Reservation updated successfully.');
```

```
47 }
48
49 public function destroy(Reservation $reservation)
50 {
51     $reservation->delete();
52
53     return redirect()->route('admin.reservations')->with('success', 'Reservation deleted successfully.');
```

Metoda update() typu POST, obsługuje walidację wprowadzonych danych w zależności od typu: pola gościa albo id użytkownika, a także datę rezerwacji. Dodaje wprowadzone zmiany do bazy oraz zwraca komunikat o sukcesie.

Metoda destroy() typu DELETE, obsługuje usuwanie rezerwacji w bazie danych. Wraz z rezerwacją usuwane są powiązane z niej bilety z tabeli 'ticket_user'. Pomyślne usunięcie zwraca nam widok z komunikatem sukcesu.

Zarządzanie biletami rezerwacji

Każda rezerwacja ma powiązane ze sobą bilety, wydzielonej do tego tablicy w bazie danych. Administrator może w pełni zarządzać w przygotowanym do tego widoku tymi biletami.

The screenshot shows a web interface titled "User Tickets" with a blue button "Add New User Ticket" in the top right. Below the title, there are two green cards representing tickets for "Reservation ID: 3".

The first card shows:

- Ticket Type: Standard
- Quantity: 2
- Buttons: Edit (yellow), Delete (red)

The second card shows:

- Ticket Type: Discount
- Quantity: 1
- Buttons: Edit (yellow), Delete (red)

Rysunek 37 - przykładowy główny widok biletów rezerwacji

The screenshot shows the "User Tickets" interface with the "Add New User Ticket" button. The main area displays a green card for "Reservation ID: 3" with "Ticket Type: Standard" and "Quantity: 2". Below this card, there is an "Edit" button (yellow) and a "Delete" button (red).

Below the buttons, there is a "Ticket" section with a dropdown menu showing "-- Select Ticket --". Below that, there is a "Quantity" section with a text input field containing the value "2". At the bottom right of the form, there are "Save" (blue) and "Cancel" (grey) buttons.

Rysunek 38 - edycja biletu dla rezerwacji nr.3

User Tickets

Add New User Ticket

Reservation ID

Ticket

-- Select Ticket --

Quantity

Add User Ticket

Rysunek 39 - formularz dodawania nowego biletu dla danej rezerwacji

Funkcje do wyświetlania i dodawania biletów do rezerwacji

```

14 public function index()
15 {
16     $user_tickets = TicketUser::with('ticket')->get();
17     $tickets = Ticket::all();
18     return view('admin.tickets_userCRUD', compact('user_tickets', 'tickets'));
19 }
20
21 public function store(Request $request)
22 {
23     $request->validate([
24         'reservation_id' => 'required|numeric',
25         'ticket_id' => 'required|numeric',
26         'quantity' => 'required|numeric|min:1',
27     ]);
28
29     $reservationExists = Reservation::where('id', $request->reservation_id)->exists();
30
31     if (!$reservationExists) {
32         throw ValidationException::withMessages([
33             'reservation_id' => 'Reservation ID not found.',
34         ]);
35     }
36
37     TicketUser::create([
38         'reservation_id' => $request->reservation_id,
39         'ticket_id' => $request->ticket_id,
40         'quantity' => $request->quantity,
41     ]);
42
43     return redirect()->route('admin.user_tickets.index')->with('success', 'User ticket added successfully!');
44 }

```

Metoda `index()` to metoda GET, która pobiera z bazy bilety dla rezerwacji oraz rodzaje biletów. Zwraca ona widok `'admin.tickets_userCRUD'`.

Metoda `store()` to metoda POST, obsługuje walidację wprowadzonych danych i wyszukuje rezerwacji o podanym id. Zwraca odpowiedni komunikat w przypadku kiedy nie zostanie znalezione podane id rezerwacji albo informacje o pomyślnym dodaniu nowego biletu dla danej rezerwacji.

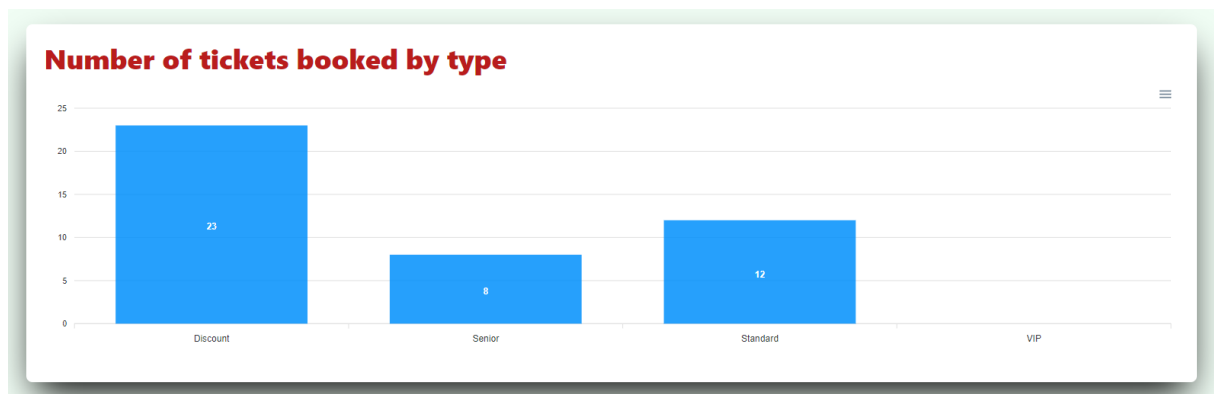
Statystyki

Aplikacja wyświetla statystkę w postaci wykresu, informuje ona administratora o łącznej rezerwacji danego typu biletu.

Wykres jest generowany dynamicznie na podstawie danych w bazie danych, dodanie nowych typów biletu oraz rezerwacji nie narusza działania wykresów.

Dostarczona jest obsługa pobrania wykresu w formatach:

- SVG
- CSV
- PNG



Rysunek 40 - wykres rezerwacji biletów wg. rodzaju