

<b>AI2</b>	<b>Dokumentacja projektu</b>
<b>Autor</b>	Piotr Rojek, 125159
<b>Kierunek, rok</b>	Informatyka, IV rok, st. stacjonarne (3,5-l)
<b>Temat projektu</b>	<i>Zarządzanie wycieczkami</i>

## Spis treści

<b>1.</b>	<b>Wstęp</b>	3
<b>1.1.</b>	<b>Użyte technologie</b>	3
<b>1.2.</b>	<b>Funkcjonalności aplikacji</b>	3
<b>2.</b>	<b>Narzędzia i technologie</b>	5
<b>2.1.</b>	<b>Baza danych</b>	5
<b>2.2.</b>	<b>Backend (Node.js i Express.js)</b>	5
<b>2.3.</b>	<b>Frontend (SPA)</b>	6
<b>3.</b>	<b>Architektura aplikacji</b>	7
<b>3.1.</b>	<b>Middleware</b>	7
<b>3.2.</b>	<b>Controllers</b>	8
<b>3.3.</b>	<b>Routes</b>	10
<b>4.</b>	<b>Baza danych</b>	11
<b>4.1.</b>	<b>Diagram ERD</b>	11
<b>4.2.</b>	<b>Krótki opis tabel</b>	11
<b>5.</b>	<b>Interfejs niezalogowanego użytkownika</b>	12
<b>5.1.</b>	<b>Strona główna</b>	12
<b>5.2.</b>	<b>Kontynenty</b>	12
<b>5.3.</b>	<b>Kraje</b>	13
<b>5.4.</b>	<b>Wycieczki</b>	15
<b>5.5.</b>	<b>Logowanie</b>	16
<b>5.6.</b>	<b>Rejestracja</b>	17
<b>6.</b>	<b>Interfejs zalogowanego użytkownika</b>	17
<b>6.1.</b>	<b>Mój profil</b>	17
<b>6.2.</b>	<b>Rezerwacja wycieczki</b>	18
<b>7.</b>	<b>Interfejs administratora</b>	19
<b>7.1.</b>	<b>Zarządzanie zasobami</b>	19
<b>7.2.</b>	<b>Użytkownicy</b>	22

<b>7.3. Rezerwacje .....</b>	23
<b>8. Endpointy API .....</b>	24
<b>8.1. Konfiguracja i struktura endpointów .....</b>	24
<b>8.2. Dokumentacja API w Swagger UI.....</b>	26
<b>8.3. Zgodność API z 2 poziomem modelu dojrzałości Richardsona .....</b>	27
<b>9. Logika biznesowa.....</b>	28
<b>9.1. Realizacja sortowania.....</b>	28
<b>9.2. Realizacja filtrowania .....</b>	30
<b>9.3. Realizacja wyszukiwania.....</b>	32
<b>9.4. Realizacja paginacji.....</b>	34
<b>10. Uruchomienie aplikacji .....</b>	36
<b>10.1. Wymagania.....</b>	36
<b>10.2. Kroki przy pierwszym uruchomieniu.....</b>	36
<b>10.3. Kroki przy kolejnym uruchomieniu .....</b>	36

## 1. Wstęp

Temat projektu obejmuje stworzenie nowoczesnej aplikacji internetowej do zarządzania wycieczkami, opartej na architekturze REST API (Representational State Transfer & Application Programming Interface) oraz jednostronicowej aplikacji klienckiej SPA (Single Page Application). System został zaprojektowany w celu ułatwienia obsługi oferty wycieczek, zarządzania rezerwacjami oraz administracji danymi geograficznymi (kontynentami i krajami) oraz użytkownikami.

Aplikacja składa się z dwóch niezależnych warstw. Backend został wykonany w technologii Node.js z użyciem frameworka Express.js, natomiast frontend został zbudowany w technologii Vue 3 i uruchamiany przy pomocy Vite. Komunikacja między warstwami odbywa się za pośrednictwem zdefiniowanych endpointów REST API, zwracających dane w formacie JSON.

W aplikacji zastosowano również obsługę uwierzytelniania i autoryzacji w oparciu o tokeny JWT (JSON Web Tokens), co pozwala na rozróżnienie ról użytkowników (administrator oraz użytkownik standardowy) oraz nadanie im odpowiednich uprawnień. Administrator posiada pełną kontrolę nad zasobami systemu, natomiast zalogowani użytkownicy mogą przeglądać wycieczki oraz dokonywać rezerwacji.

### 1.1. Użyte technologie

- **Backend (API):**
  - Node.js 22.x.
  - Express.js 5.x.
  - Baza danych: SQLite 3 (obsługiwana poprzez bibliotekę sqlite3).
  - Uwierzytelnianie: JWT (JSON Web Tokens), bcrypt.
  - Middleware: Express, CORS, Body-parser.
  - Dokumentacja API: Swagger (OpenAPI 3.0).
- **Frontend (SPA):**
  - Vue.js 3.x.
  - Vite 7.x.
  - Pinia (magazyn stanu aplikacji).
  - Vue Router 4.x (obsługa nawigacji i guardów autoryzacyjnych).
  - HTML5, CSS3, JavaScript (ES2020+).

### 1.2. Funkcjonalności aplikacji

Aplikacja udostępnia zestaw funkcji zależnych od poziomu dostępu użytkownika (gość, użytkownik zalogowany, administrator):

#### 1. Zasoby ogólnodostępne:

- Dostęp do strony głównej, listy wycieczek, kontynentów oraz krajów jest dostępny dla każdego użytkownika, niezależnie od tego, czy jest on zalogowany.
- Widok szczegółów wycieczki, kontynentu i kraju również jest publiczny.
- Strony logowania oraz rejestracji są dostępne wyłącznie dla niezalogowanych użytkowników.

## **2. Rejestracja i logowanie użytkowników:**

- Nowi użytkownicy mogą samodzielnie się zarejestrować poprzez formularz rejestracji. Po pomyślnej rejestracji użytkownik może zalogować się na swoje konto.
- Dostęp do pełnych funkcji aplikacji, takich jak składanie rezerwacji, przeglądanie własnych rezerwacji czy aktualizacja danych osobowych, mają wyłącznie zarejestrowani i zalogowani użytkownicy.

## **3. Funkcjonalności dla zalogowanego użytkownika:**

- Użytkownik ma dostęp do wszystkich funkcji dostępnych dla niezalogowanego gościa.
- Może dokonywać rezerwacji wycieczki, na którą nie jest zapisany, poprzez odpowiedni przycisk w widoku szczegółowym wybranej wycieczki.
- Posiada dostęp do zakładki „Mój Profil”, gdzie może przeglądać oraz edytować swoje dane osobowe (imię, nazwisko, adres e-mail, hasło).

## **4. Funkcjonalności dla zalogowanego administratora:**

- Administrator posiada najwyższy poziom uprawnień i może wykonywać pełne operacje CRUD na wszystkich zasobach systemu (użytkownicy, wycieczki, kraje, kontynenty, rezerwacje).
- W widoku szczegółowym rezerwacji administrator może zmieniać status rezerwacji użytkowników.
- Administrator może edytować wszystkie zasoby systemu. Dodatkowo może usuwać rekordy, jeśli taka operacja nie narusza spójności pozostałych danych w systemie.
- Posiada pełny dostęp do całej aplikacji.

## **5. Automatyzacja procesów biznesowych:**

- System automatycznie weryfikuje poprawność tokenu JWT oraz uprawnienia użytkownika przy każdej próbie wykonania operacji chronionej.
- Walidacja danych wejściowych jest przeprowadzana zarówno po stronie frontendu, jak i backendu.
- Przed usunięciem kontynentu, kraju, wycieczki lub użytkownika system zawsze sprawdza powiązania w bazie i blokuje operację, jeśli ona może naruszyć spójność danych.
- Rezerwacje użytkownika są automatycznie powiązane z kontem zalogowanego użytkownika. Nie ma możliwości dokonania rezerwacji jako inna osoba.
- Lista wszystkich zasobów jest dynamicznie aktualizowana dzięki wykorzystaniu filtrowania, sortowania, wyszukiwania oraz paginacji (wszystko obsługiwane po stronie API).
- Każdy moduł posiada ujednolicone endpointy API, co ułatwia rozszerzanie aplikacji o kolejne funkcjonalności.

## 2. Narzędzia i technologie

### 2.1. Baza danych

W projekcie zastosowano lekki i wydajny silnik bazodanowy SQLite 3, który idealnie sprawdza się w aplikacjach backendowych wykorzystujących REST API. SQLite jest bazą danych działającą w trybie plikowym, czyli wszystkie dane przechowywane są w jednym pliku. Dzięki temu konfiguracja jest znacznie prostsza niż w przypadku klasycznych serwerów bazodanowych, takich jak PostgreSQL czy MySQL.

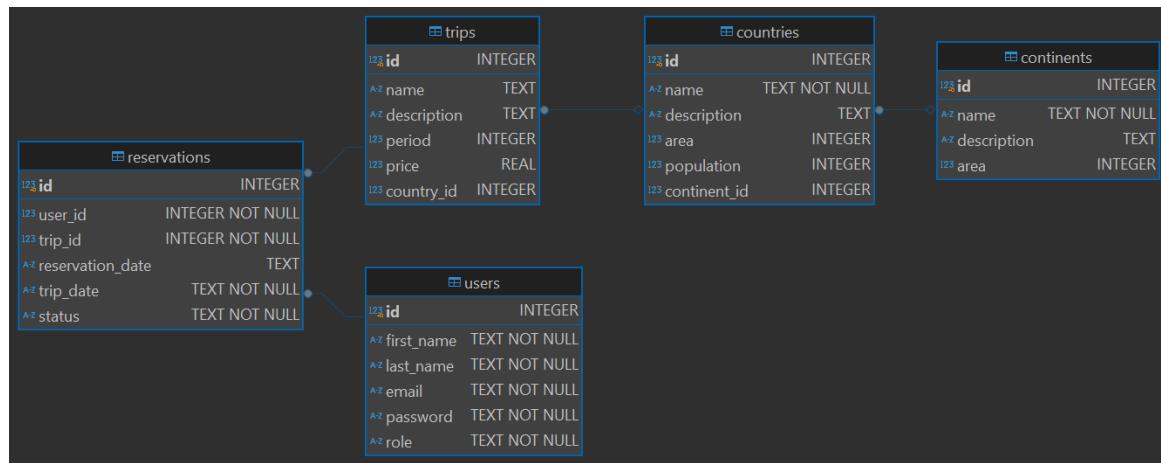
SQLite jest oprogramowaniem open-source i nie wymaga żadnej instalacji serwera. Jednak do wygodnego przeglądania danych można użyć programu DBeaver, który można pobrać ze strony <https://dbeaver.io/download/>. Biblioteka `sqlite3` używana przez backend pozwala wykonywać operacje SQL bezpośrednio z poziomu aplikacji Node.js. Bibliotekę oraz dokumentację można znaleźć na stronie <https://www.sqlite.org/download.html>.

Konfiguracja połączenia z bazą odbywa się w pliku `/db/database.js`, gdzie aplikacja:

- Tworzy nowe połączenie z bazą.
- Włącza obsługę kluczy obcych.
- Tworzy wymagane tabele, jeśli jeszcze nie istnieją.
- Uzupełnia tabele odpowiednimi danymi startowymi.

Nie ma potrzeby edycji pliku `.env` w zakresie konfiguracji bazy, ponieważ SQLite nie wymaga podawania loginu, hasła ani adresu serwera. Jedyną opcją, którą można zmienić, jest ścieżka do pliku `.db`, gdyby w przyszłości zaszła potrzeba przeniesienia bazy danych.

Diagram ERD bazy danych widoczny jest poniżej.



### 2.2. Backend (Node.js i Express.js)

Backend aplikacji został stworzony z wykorzystaniem środowiska Node.js w wersji 22.x. Node jest platformą, która umożliwia uruchamianie kodu JavaScript poza przeglądarką, bezpośrednio na serwerze. Dzięki temu można tworzyć aplikacje serwerowe w tym samym języku, w którym pisany jest frontend.

Warstwa serwerowa korzysta z frameworka Express.js w wersji 5.x. Express jest lekkim i elastycznym narzędziem, które umożliwia szybkie tworzenie API oraz obsługę żądań HTTP. W projekcie pełni on główną rolę w obsłudze logiki biznesowej i komunikacji z bazą danych.

Backend korzysta z kilku kluczowych bibliotek. Biblioteka sqlite3 umożliwia łączenie się z bazą danych SQLite oraz wykonywanie operacji SQL z poziomu Node.js. Biblioteka bcrypt służy do bezpiecznego hashowania haseł użytkowników przed zapisaniem ich w bazie danych. Zastosowano również moduł CORS, który kontroluje dostęp do API z poziomu przeglądarki oraz zabezpiecza komunikację między frontendem a backendem. Biblioteka Body-parser umożliwia poprawne odczytywanie danych JSON wysyłanych w zapytaniach HTTP przez frontend.

W projekcie zastosowano również mechanizm dokumentacji API oparty na Swagger UI oraz standardzie OpenAPI 3.0. Dokumentacja dostępna pod adresem /api-docs umożliwia przeglądanie wszystkich dostępnych endpointów. Swagger generuje interaktywny interfejs, który pozwala testować działanie API bez konieczności używania zewnętrznych narzędzi, takich jak Postman.

Backend odpowiada za całą logikę biznesową aplikacji. Obejmuje to między innymi obsługę operacji CRUD dla kontynentów, krajów, wycieczek, użytkowników oraz rezerwacji. Warstwa serwerowa zarządza również procesem uwierzytelniania i autoryzacji użytkowników z użyciem tokenów JWT. Backend waliduje dane wejściowe i sprawdza ich poprawność jeszcze przed zapisaniem ich do bazy.

### 2.3. Frontend (SPA)

Frontend aplikacji został zbudowany jako jednostronnicowa aplikacja kliencka (SPA). Do jego stworzenia wykorzystano framework Vue.js w wersji 3.x, który opiera się na nowoczesnym podejściu Composition API. Umożliwia to tworzenie modularnych, dynamicznych i przejrzystych interfejsów użytkownika.

Do uruchamiania projektu frontendowego użyto narzędzia Vite w wersji 7.x. Vite znacznie przyspiesza proces programowania dzięki natychmiastowemu przeładowaniu zmian w kodzie oraz optymalnemu budowaniu aplikacji na produkcję.

W projekcie wykorzystano również magazyn stanu Pinia. Pinia jest oficjalnym rozwiązaniem dla Vue.js, które pozwala na przechowywanie danych globalnych, na przykład informacji o zalogowanym użytkowniku i jego tokenie. Centralizacja danych ułatwia zarządzanie stanem oraz synchronizację danych między komponentami.

Do obsługi nawigacji w aplikacji zastosowano Vue Router w wersji 4.x. Router odpowiada za przełączanie widoków i renderowanie odpowiednich komponentów w zależności od ścieżki URL.

Frontend korzysta ze standardowych technologii internetowych, takich jak HTML5, CSS3 i JavaScript. Te technologie służą do tworzenia struktury strony, stylowania elementów oraz obsługi interakcji użytkownika.

Frontend działa całkowicie niezależnie od backendu. Oznacza to, że obie warstwy mogą być rozwijane osobno, a komunikacja między nimi odbywa się wyłącznie poprzez REST API. Jest to nowoczesne podejście zgodne z architekturą typu klient-serwer, zapewniające elastyczność i łatwość rozbudowy aplikacji.

## 3. Architektura aplikacji

### 3.1. Middleware

Middleware to pośrednie funkcje, które obsługują żądania HTTP przed ich przekazaniem do kontrolerów. Umożliwiają filtrowanie i modyfikowanie żądań oraz odpowiedzi, co pozwala na wprowadzanie dodatkowej logiki na różnych etapach przetwarzania żądań.

Główne funkcje middleware:

- Autoryzacja i uwierzytelnianie: Mogą sprawdzać, czy użytkownik jest zalogowany i czy ma odpowiednie uprawnienia do wykonania danej akcji.
- Kontrola dostępu: Mogą blokować wybrane trasy w zależności od roli użytkownika lub stanu sesji.
- Modyfikacja żądań i odpowiedzi: Mogą modyfikować zawartość żądań przed ich przekazaniem do kontrolera oraz zmieniać odpowiedzi przed ich wysłaniem do klienta.

W projekcie występują cztery middleware:

**1. verifyToken:**

- Sprawdza, czy nagłówek Authorization zawiera poprawny token JWT.
- Ustawia w req.user dane zalogowanego użytkownika.
- W razie błędu zwraca status 401 lub 403.

**2. isAdmin:**

- Zezwala na dostęp tylko użytkownikom z rolą admin.
- W przypadku braku uprawnień zwraca status 403.

**3. isAdminOrSelf:**

- Stosowany przy zasobach związanych z profilem użytkownika.
- Pozwala użytkownikowi na dostęp tylko do własnych danych lub przyznaje pełny dostęp administratorowi. W przeciwnym razie zwraca status 403.

#### 4. checkNotAuthenticated:

- Blokuje ponowny dostęp do logowania oraz rejestracji, jeśli użytkownik jest już zalogowany.

Przykład middleware, wykorzystanego w projekcie, został przedstawiony poniżej.

```
1 // Import bibliotek
2 const jwt = require("jsonwebtoken");
3 require("dotenv").config();
4
5 // Klucz JWT
6 const JWT_SECRET = process.env.JWT_SECRET || 'domyslny_tajny_klucz';
7
8 // Middleware sprawdzajcy, czy uzytkownik jest zalogowany
9 exports.verifyToken = (req, res, next) => {
10   const authHeader = req.headers['authorization'];
11   const token = authHeader && authHeader.split(' ')[1];
12   if (!token) return res.status(401).json({ error: 'Brak tokenu autoryzacyjnego.' });
13
14   jwt.verify(token, JWT_SECRET, (err, user) => {
15     if (err) return res.status(403).json({ error: 'Nieprawidlowy lub wygasly token autoryzacyjny.' });
16     req.user = user;
17     next();
18   });
19 };
20
21 // Funkcja pomocnicza sprawdzajaca, czy uzytkownik jest administratorem
22 function isAdmin(user) {
23   return user && user.role === 'admin';
24 }
25
26 // Middleware sprawdzajcy, czy uzytkownik ma role administratora
27 exports.isAdmin = (req, res, next) => {
28   if (!isAdmin(req.user)) return res.status(403).json({ error: 'Nie posiadasz uprawnien administratora.' });
29   next();
30 };
31
32 // Middleware sprawdzajcy, czy uzytkownik ma role administratora lub odwoluje sie do swego ID
33 exports.isAdminOrSelf = (req, res, next) => {
34   const loggedUser = req.user;
35   const requestedUserId = Number(req.params.userId);
36   if (isAdmin(loggedUser) || loggedUser.id === requestedUserId) return next();
37   return res.status(403).json({ error: 'Nie posiadzasz uprawnien do tych danych.' });
38 };
```

### 3.2. Controllers

Kontrolery są kluczowym elementem, gdyż obsługują logikę aplikacji po stronie backendu. Służą one do przetwarzania żądań HTTP, wykonywania operacji na bazie danych i zwracania odpowiedzi w formacie JSON. W tym projekcie kontrolery pełnią również funkcję warstwy modelu, ponieważ zapytania SQL są wykonywane bezpośrednio w kontrolerach.

Główne funkcje kontrolerów:

- Obsługa żądań HTTP: Kontrolery przyjmują żądania HTTP (GET, POST, PUT, DELETE) i odpowiadają na nie odpowiednimi danymi. Każda metoda w kontrolerze odpowiada na konkretne żądanie.
- Zwracanie odpowiedzi JSON: Backend przekazuje dane w formacie JSON, z którego korzysta frontend.
- Walidacja danych: Kontrolery mogą przeprowadzać walidację danych przychodzących z żądań, zapewniając, że dane są prawidłowe przed ich przetworzeniem lub zapisaniem do bazy danych.
- Struktura i organizacja kodu: Kontrolery pomagają w organizacji kodu, dzieląc aplikację na logiczne segmenty.

W projekcie znajduje się sześć kontroletów:

#### 1. usersController:

- Zarządza użytkownikami systemu.
- Zawiera funkcje do pobierania i aktualizowania profilu zalogowanego użytkownika.

## 2. continentsController:

- Obejmuje pełną obsługę kontynentów.

## 3. countriesController:

- Zarządza operacjami związanymi z krajami.

## 4. tripsController:

- Odpowiada za wszystkie operacje na wycieczkach.

## 5. reservationsController:

- Obsługuje rezerwacje wycieczek.
- Zawiera logikę liczenia rezerwacji dla danego użytkownika lub wycieczki oraz funkcję umożliwiającą użytkownikowi dokonanie rezerwacji.

## 6. authController:

- Zarządza uwierzytelnianiem użytkowników, obejmując rejestrację, logowanie i weryfikację tokenu JWT.

Przykład fragmentu kontrolera, wykorzystanego w projekcie, został przedstawiony poniżej.

```
1 // Import połączenia z bazą danych
2 const db = require('../db/database');
3
4 // Import bibliotek
5 const bcrypt = require('bcrypt');
6 const jwt = require('jsonwebtoken');
7 require('dotenv').config();
8
9 // Klucz JWT
10 const JWT_SECRET = process.env.JWT_SECRET || 'domyslny_tajny_klucz';
11
12 // Pомocnicza funkcja do walidacji i standaryzacji adresu e-mail
13 function validateAndNormalizeEmail(email) {
14     if (!email) return null;
15     const normalized = email.trim().toLowerCase();
16     const emailRegex = /^[a-zA-Z\d+-.]+@[a-zA-Z\d+\-]+\.[a-zA-Z]{2,}\$/;
17     if (!emailRegex.test(normalized)) return null;
18     return normalized;
19 }
20
21 // Rejestracja użytkownika
22 exports.register = (req, res) => {
23     // Pobranie parametrów przesyłanych w ciele żądania
24     const { first_name, last_name, email, password, role = 'user' } = req.body;
25
26     // Walidacja wymaganych danych
27     if (!first_name || !last_name || !email || !password) return res.status(400).json({ error: 'Brak wymaganych danych.' });
28
29     // Walidacja typu danych
30     if (![ 'admin', 'user' ].includes(role)) return res.status(400).json({ error: 'Nieprawidłowa rola użytkownika.' });
31
32     const normalizedEmail = validateAndNormalizeEmail(email);
33     if (!normalizedEmail) return res.status(400).json({ error: 'Nieprawidłowy adres e-mail.' });
34
35     // Wykonanie zapytania do pobrania użytkownika w celu sprawdzenia duplikatu
36     db.get(
37         'SELECT id FROM users WHERE email = ?',
38         [normalizedEmail],
39         (err, existingUser) => {
40             if (err) return res.status(500).json({ error: 'Błąd serwera podczas pobierania użytkownika.' });
41             if (existingUser) return res.status(409).json({ error: 'Użytkownik o podanym adresie e-mail już istnieje.' });
42
43             // Hashowanie hasła
44             const hashedPassword = bcrypt.hashSync(password, 10);
45
46             // Wykonanie polecenia do rejestracji użytkownika
47             db.run(
48                 'INSERT INTO users (first_name, last_name, email, password, role) VALUES (?, ?, ?, ?, ?)',
49                 [first_name, last_name, normalizedEmail, hashedPassword, role],
50                 function (err) {
51                     if (err) return res.status(500).json({ error: 'Błąd serwera podczas rejestracji użytkownika.' });
52
53                     // Tworzenie tokenu JWT
54                     const token = jwt.sign(
55                         { id: this.lastID, email: normalizedEmail, role },
56                         JWT_SECRET,
57                         { expiresIn: '2h' }
58                     );
59
60                     res.status(201).json({ message: 'Użytkownik zarejestrowany.', user: { id: this.lastID, first_name, last_name, email, role }, token });
61                 }
62             );
63         }
64     );
65 }
```

### 3.3. Routes

Każdy moduł aplikacji posiada osobny plik, który jest używany do definiowania tras. Trasy mapują konkretne adresy URL na odpowiednie funkcje kontrolerów i określają, które middleware muszą zostać uruchomione przed wykonaniem danej operacji.

Główne funkcje routes:

- Mapowanie adresów: Każdy plik routingu przypisuje konkretne ścieżki do odpowiednich funkcji kontrolerów.
- Przypisywanie middleware: Trasy wymagające autoryzacji czy uprawnień administratora są odpowiednio zabezpieczane przez middleware.

W projekcie znajduje się sześć routingów:

**1. users:**

- Obsługuje operacje na użytkownikach oraz funkcje z profilem zalogowanego użytkownika.

**2. continents:**

- Zawiera trasy dotyczące kontynentów.

**3. countries:**

- Obsługuje operacje związane z krajami.

**4. trips:**

- Odpowiada za mapowanie tras dla wycieczek.

**5. reservations:**

- Obejmuje trasy związane z rezerwacjami wycieczek.

**6. auth:**

- Zawiera trasy logowania, rejestracji oraz weryfikacji tokenów JWT.

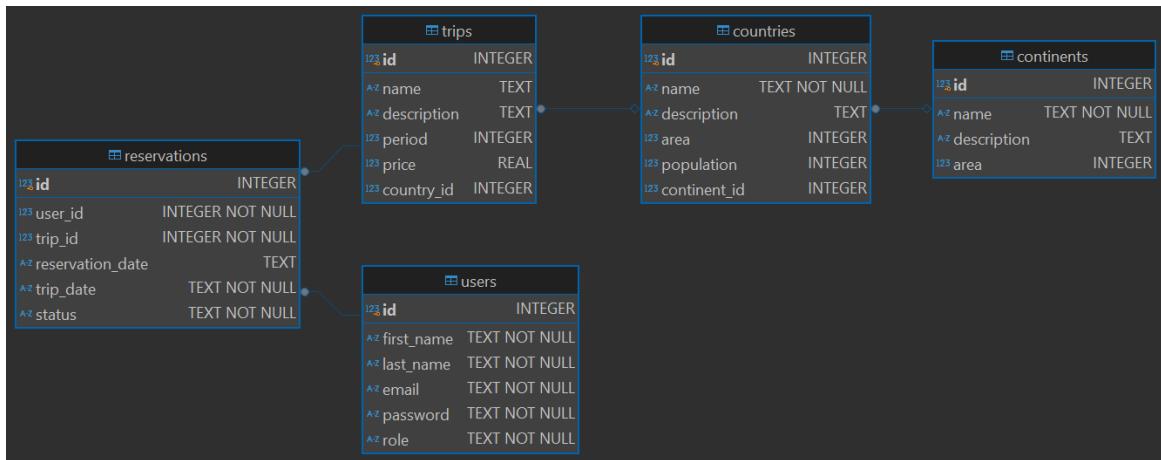
Przykład pliku routes, zastosowanego w projekcie, został przedstawiony poniżej.

```
1 // Import bibliotek
2 const express = require('express');
3 const router = express.Router();
4
5 // Import middlewareów
6 const { verifyToken, isAdmin, isAdminOrSelf } = require('../middleware/authMiddleware');
7
8 // Import kontrolera z logiką obsługą endpointów
9 const controller = require('../controllers/reservationsController');
10
11 // Definicja endpointów i powiązanie ich z funkcjami kontrolera
12
13 /**
14 * @swagger
15 * tags:
16 *   name: Reservations
17 *   description: Operacje na rezerwacjach
18 */
19
20 /**
21 * @swagger
22 * /reservations/count/trip/{trip_id}:
23 *   get:
24 *     summary: Pobieranie liczby rezerwacji wycieczki
25 *     tags: [Reservations]
26 *     parameters:
27 *       - in: path
28 *         name: trip_id
29 *         required: true
30 *         schema:
31 *           type: integer
32 *           description: ID wycieczki
33 *     responses:
34 *       200:
35 *         description: Pobrano liczbę rezerwacji wycieczki.
36 *       400:
37 *         description: Nieprawidłowy ID wycieczki.
38 *       404:
39 *         description: Wycieczka o podanym ID nie istnieje.
40 *       500:
41 *         description: Błąd serwera podczas pobierania liczby rezerwacji wycieczki.
42 */
43 router.get('/count/trip/:trip_id', controller.getReservationsCountByTrip);
```

## 4. Baza danych

### 4.1. Diagram ERD

Diagram ERD wygenerowany z bazy danych, odnoszący się do struktur tabel zastosowanych w projekcie, został przedstawiony poniżej.



### 4.2. Krótki opis tabel

- **Tabela „continents”**: Przechowuje informacje o kontynentach, takie jak nazwa kontynentu (name), jego opis (description) oraz powierzchnia (area).
- **Tabela „countries”**: Zawiera dane dotyczące krajów, takie jak nazwa (name), opis (description), powierzchnia (area), populacja (population) oraz klucz obcy continent\_id wskazujący kontynent, do którego dany kraj należy.
- **Tabela „trips”**: Przechowuje dane wycieczek, w tym ich nazwę (name), opis (description), czas trwania (period), cenę (price) oraz klucz obcy country\_id wskazujący kraj, z którym wycieczka jest powiązana.
- **Tabela „users”**: Zawiera dane użytkowników aplikacji, takie jak imię (first\_name), nazwisko (last\_name), adres e-mail (email), hasło (password) oraz rola użytkownika (role), określająca uprawnienia.
- **Tabela „reservations”**: Przechowuje informacje o rezerwacjach wycieczek. Zawiera klucze obce user\_id i trip\_id, datę złożenia rezerwacji (reservation\_date), datę planowanej wycieczki (trip\_date) oraz status rezerwacji (status).

## 5. Interfejs niezalogowanego użytkownika

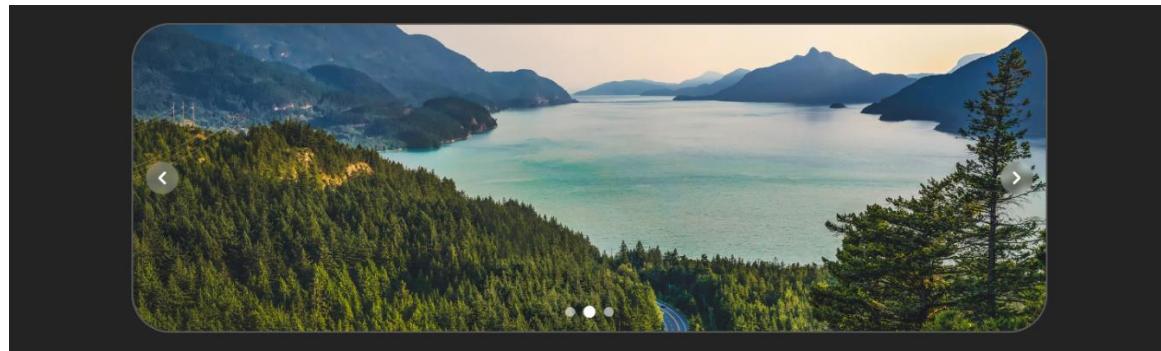
### 5.1. Strona główna

Strona główna aplikacji "URocze wycieczki" jest pierwszym ekranem widocznym po wejściu do aplikacji. Pełni ona funkcję informacyjną oraz nawigacyjną, kierując użytkownika do najważniejszych sekcji systemu.

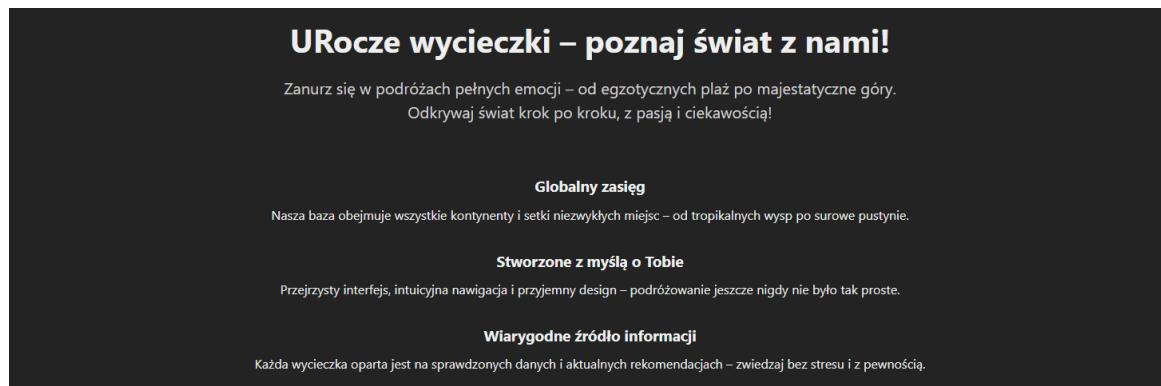
- Nagłówek:** Zawiera logo aplikacji (kulę ziemską) umieszczoną po lewej stronie. Posiada odnośniki do innych podstron. Zawiera też linki do sekcji logowania i rejestracji użytkowników po prawej stronie. Posiada również opcję zmiany motywu strony.



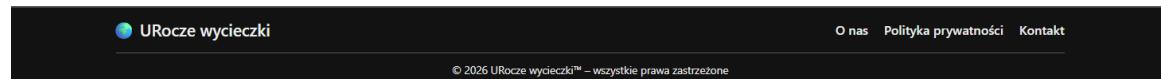
- Karuzela:** Dynamiczny slider zawierający przykładowe zdjęcia z wycieczek.



- Sekcja powitalna:** Prosty ekran wprowadzający użytkownika do tematyki aplikacji, zachęcający do przeglądania dostępnych wycieczek.



- Stopka:** Zawiera proste informacje oraz linki do innych podstron.



### 5.2. Kontynenty

Widok wyświetla wszystkie kontynenty zapisane w bazie danych w postaci tabeli. Strona wspiera wyszukiwanie, sortowanie i paginację danych. Każdy kontynent można otworzyć, aby zobaczyć jego szczegóły.

[URocze wycieczki](#)

Wycieczki Kraje Kontynenty

Logowanie Rejestracja ☀

## Kontynenty świata

Poznaj każdy z kontynentów – odkrywaj świat krok po kroku i zobacz lądy, które tworzą mozaikę naszej planety.

3 kontynenty na stronę ▾

Kontynent	Powierzchnia (km <sup>2</sup> )	Link
Europa	10 180 000	<a href="#">Szczegóły</a>
Azja	44 579 000	<a href="#">Szczegóły</a>
Australia	7 692 000	<a href="#">Szczegóły</a>

[← Poprzednia](#)
Strona 1 z 3
[Następna →](#)

---

[URocze wycieczki](#)

O nas Polityka prywatności Kontakt

© 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone

[URocze wycieczki](#)

Wycieczki Kraje Kontynenty

Logowanie Rejestracja ☀

## Szczegóły kontynentu

Poznaj informacje o wybranym kontynencie.



**Australia**

Najmniejszy kontynent, znany z unikalnej fauny.

Powierzchnia: 7 692 000 km<sup>2</sup>

Liczba krajów: 9

[Powrót do listy](#)

[← Poprzedni](#)
[Następny →](#)

---

[URocze wycieczki](#)

O nas Polityka prywatności Kontakt

© 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone

### 5.3. Kraje

Strona wyświetla wszystkie kraje dostępne w systemie, wraz z możliwością przeglądania ich powiązań z kontynentami. Widok obsługuje wyszukiwanie, sortowanie, filtrowanie po kontynencie oraz paginację danych. Dodatkowo każdy kraj zawiera link przenoszący do szczegółowego widoku danego kraju.

## Dostępne kraje

Zanurz się w różnorodności kultur i krajobrazów. Nasze wycieczki obejmują kraje z każdego zakątku świata.

Szukaj kraju lub kontynentu...

Wszystkie kontynenty

10 krajów na stronę

Kraj	Kontynent	Powierzchnia (km <sup>2</sup> )	Ludność	Link
Polska	Europa	312 679	38 000 000	Szczegóły
Łotwa	Europa	64 589	1 900 000	Szczegóły
Estonia	Europa	45 227	1 300 000	Szczegóły
Dania	Europa	43 094	5 800 000	Szczegóły
Szwecja	Europa	450 295	10 300 000	Szczegóły
Czechy	Europa	78 865	10 700 000	Szczegóły
Austria	Europa	83 879	8 900 000	Szczegóły
Słowacja	Europa	49 037	5 400 000	Szczegóły
Szwajcaria	Europa	41 285	8 600 000	Szczegóły
Hiszpania	Europa	505 990	47 000 000	Szczegóły

← Poprzednia

Strona 1 z 14

Następna →

## Szczegóły kraju

Poznaj informacje o wybranym kraju.



### Polka

Kraj w Europie Wschodniej.

**Kontynent:** Europa

**Powierzchnia:** 312 679 km<sup>2</sup>

**Ludność:** 38 000 000

**Liczba wycieczek:** 5

[Powrót do listy](#)

← Poprzedni

Następny →

## 5.4. Wycieczki

Widok przedstawia pełną listę wycieczek w formie tabeli, z możliwością filtrowania po kontynencie i kraju, wyszukiwania, sortowania i paginacji. Użytkownik po kliknięciu w odpowiedni link może otworzyć szczegóły wybranej wycieczki.

URocze wycieczki Logowanie Rejestracja ☀

### Nasze wycieczki

Odkryj świat z nami – od egzotycznych plaż po majestatyczne góry. Wybierz kierunek, który rozbudzi Twoją ciekawość!

Szukaj wycieczki, kontynentu lub kraju...

Wszystkie kontynenty ▼ Wszystkie kraje ▼ 20 wycieczek na stronę ▼

Nazwa wycieczki	Kontynent	Kraj	Okres	Cena	Link
Wakacje w Krakowie	Europa	Polska	3 dni	900 PLN	<a href="#">Szczegóły</a>
Trójmiasto	Europa	Polska	5 dni	1500 PLN	<a href="#">Szczegóły</a>
Nieznany Kair	Afryka	Egipt	4 dni	1600 PLN	<a href="#">Szczegóły</a>
Safari w Egipcie	Afryka	Egipt	5 dni	2700 PLN	<a href="#">Szczegóły</a>
Wodny Asuan	Afryka	Egipt	5 dni	2200 PLN	<a href="#">Szczegóły</a>
Wyprawa do Tokio	Azja	Japonia	7 dni	4000 PLN	<a href="#">Szczegóły</a>
Starożytny Luksor	Afryka	Egipt	4 dni	2000 PLN	<a href="#">Szczegóły</a>
NieOsaka	Azja	Japonia	5 dni	3000 PLN	<a href="#">Szczegóły</a>
Toksyczna Hiroshima	Azja	Japonia	4 dni	2800 PLN	<a href="#">Szczegóły</a>
Zwiedzanie Kioto	Azja	Japonia	6 dni	3500 PLN	<a href="#">Szczegóły</a>
Wyprawa do Rzymu	Europa	Włochy	5 dni	3000 PLN	<a href="#">Szczegóły</a>
Pływająca Wenecja	Europa	Włochy	4 dni	2800 PLN	<a href="#">Szczegóły</a>
Florencja	Europa	Włochy	4 dni	2700 PLN	<a href="#">Szczegóły</a>
Neapol i Pompeje	Europa	Włochy	5 dni	3200 PLN	<a href="#">Szczegóły</a>
Mafijna Sycylia	Europa	Włochy	8 dni	5500 PLN	<a href="#">Szczegóły</a>
Wyprawa do Paryża	Europa	Francja	5 dni	3200 PLN	<a href="#">Szczegóły</a>
Luwr	Europa	Francja	3 dni	2000 PLN	<a href="#">Szczegóły</a>
Dolina Loary	Europa	Francja	4 dni	2500 PLN	<a href="#">Szczegóły</a>
Lazurowe Wybrzeże	Europa	Francja	6 dni	4000 PLN	<a href="#">Szczegóły</a>
Wyprawa do Nowego Jorku	Ameryka Północna	Stany Zjednoczone	10 dni	7900 PLN	<a href="#">Szczegóły</a>

[← Poprzednia](#) Strona 1 z 4 [Następna →](#)

URocze wycieczki O nas Polityka prywatności Kontakt

© 2026 URocze wycieczki™ – wszystkie prawa zastrzezone

URocze wycieczki

Wycieczki Kraje Kontynenty

Logowanie Rejestracja ☀

## Szczegóły wycieczki

Poznaj wszystkie informacje o wybranej podróży.



### Lazurowe Wybrzeże

Piękne plaże i luksusowe kurorty.

**Kontynent:** Europa  
**Kraj:** Francja  
**Okres:** 6 dni  
**Cena:** 4000 PLN  
**Liczba rezerwacji:** 0

[Powrót do listy](#)

[← Poprzednia](#) [Następna →](#)

URocze wycieczki

O nas Polityka prywatności Kontakt

© 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone

## 5.5. Logowanie

Widok ten umożliwia użytkownikom zalogowanie się do systemu. Zawiera pola do wpisania adresu e-mail oraz hasła. Na dole strony znajduje się link kierujący do formularza rejestracyjnego dla nowych użytkowników.

URocze wycieczki

Wycieczki Kraje Kontynenty

Logowanie Rejestracja ☀

## Logowanie

Adres e-mail

Hasło

[Zaloguj się](#)

Nie masz konta? [Zarejestruj się](#)

URocze wycieczki

O nas Polityka prywatności Kontakt

© 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone

## 5.6. Rejestracja

Strona ta umożliwia nowym użytkownikom utworzenie konta w systemie. Formularz rejestracyjny składa się z pól do wypełnienia, takich jak imię, nazwisko, adres e-mail, hasło i powtórz hasło. Na dole strony znajduje się link kierujący do formularza logowania.

The screenshot shows the registration page of a travel website. At the top, there's a navigation bar with links for 'Wycieczki', 'Kraje', 'Kontynenty', 'Logowanie' (Login), and 'Rejestracja' (Registration). The 'Rejestracja' link is underlined, indicating it's the active page. The main title 'Rejestracja' is centered at the top of the page. Below it is a large blue rectangular form area containing five input fields: 'Imię' (Name) with 'Jan' entered, 'Nazwisko' (Surname) with 'Kowalski' entered, 'Adres e-mail' (Email Address) with 'userX@example.com' entered, 'Hasło' (Password) with 'user123' entered, and 'Powtóż hasło' (Repeat Password) with 'user123' entered. Below these fields is a blue button labeled 'Zarejestruj się' (Register). At the bottom of the form, there's a link 'Masz już konto? Zaloguj się' (Already have an account? Log in). The footer of the page includes the website logo 'URocze wycieczki', links for 'O nas' (About us), 'Polityka prywatności' (Privacy policy), and 'Kontakt' (Contact), and a copyright notice: '© 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone'.

## 6. Interfejs zalogowanego użytkownika

Po pomyślnym zalogowaniu się do aplikacji, użytkownik uzyskuje dodatkowe możliwości jakie może wykonać w systemie.

Przykładowe dane do zalogowania się:

- **Email:** user1@example.com
- **Hasło:** user123

### 6.1. Mój profil

Strona ta prezentuje podstawowe informacje o użytkowniku. Są to imię i nazwisko, adres e-mail oraz rola. Dodatkowo użytkownik może edytować swoje dane, po wcisnięciu odpowiedniego przycisku. Jeśli użytkownik zarezerwował kiedykolwiek jakąś wycieczkę, widzi informację o tym w swoim profilu.



## Mój profil

**Jan Kowalski****Email:** user1@example.com  
**Rola:** Użytkownik[Zmień dane](#)**Imię**

Jan

**Nazwisko**

Kowalski

**E-mail**

user1@example.com

[Zmień hasło](#)[Zapisz zmiany](#)

### Moje rezerwacje (3)

**Nazwa wycieczki:** Wyprawa do Tokio  
**Data wycieczki:** 2025-10-10  
**Data rezerwacji:** 2025-10-09  
**Status:** Zakończony**Nazwa wycieczki:** Wakacje w Krakowie  
**Data wycieczki:** 2025-11-28  
**Data rezerwacji:** 2025-11-16  
**Status:** Oczekujący**Nazwa wycieczki:** Wyprawa do Pekinu  
**Data wycieczki:** 2025-12-02  
**Data rezerwacji:** 2025-11-16  
**Status:** Oczekujący

## 6.2. Rezerwacja wycieczki

Szczegółowy widok wycieczki pozwala zalogowanemu użytkownikowi dokonać rezerwacji danej wycieczki poprzez kliknięcie jednego przycisku. Po podaniu daty wycieczki system automatycznie wiąże rezerwację z kontem użytkownika, a w przypadku prób wielokrotnej rezerwacji tej samej wycieczki blokuje operację. Po zatwierdzeniu użytkownik otrzymuje potwierdzenie rezerwacji w formie toastu.

URocze wycieczki

Wycieczki Kraje Kontynenty Mój profil Wylogowanie ☀

## Szczegóły wycieczki

Poznaj wszystkie informacje o wybranej podróży.



### NieOsaka

Kuchnia i kultura w sercu Japonii.

**Kontynent:** Azja  
**Kraj:** Japonia  
**Okres:** 5 dni  
**Cena:** 3000 PLN  
**Liczba rezerwacji:** 1

[Powrót do listy](#) [Zarezerwuj](#)

[← Poprzednia](#) [Następna →](#)

© 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone

## 7. Interfejs administratora

Po pomyślnym zalogowaniu się do aplikacji, administrator zyskuje szereg dodatkowych funkcjonalności jakie może wykonać.

Przykładowe dane do zalogowania się:

- Email:** admin@example.com
- Hasło:** admin123

### 7.1. Zarządzanie zasobami

Administrator ma możliwość pełnego zarządzania wszystkimi zasobami systemu, takimi jak kontynenty, kraje, wycieczki, użytkownicy czy rezerwacje. W każdym widoku z tabelą zawierającą odpowiednie dane zyskuje dodatkowy przycisk, który umożliwia dodawanie nowych rekordów poprzez odpowiednie formularze. Dodatkowo w szczegółowych widokach zyskuje możliwość edytowania danych lub usuwania istniejących danych. Dzięki temu administrator kontroluje całą strukturę i zawartość aplikacji, zapewniając spójność oraz aktualność danych.

 URocze wycieczki

Wycieczki [Kraje](#) Kontynenty Użytkownicy Rezerwacje Mój profil Wylogowanie ☀

## Dostępne kraje

Zanurz się w różnorodności kultur i krajobrazów. Nasze wycieczki obejmują kraje z każdego zakątku świata.

[Dodaj kraj](#)

Szukaj kraju lub kontynentu...

Wszystkie kontynenty ▾ 5 krajów na stronę ▾

Kraj	Kontynent	Powierzchnia (km <sup>2</sup> )	Ludność	Link
Polska	Europa	312 679	38 000 000	<a href="#">Szczegóły</a>
Łotwa	Europa	64 589	1 900 000	<a href="#">Szczegóły</a>
Estonia	Europa	45 227	1 300 000	<a href="#">Szczegóły</a>
Dania	Europa	43 094	5 800 000	<a href="#">Szczegóły</a>
Szwecja	Europa	450 295	10 300 000	<a href="#">Szczegóły</a>

← Poprzednia Strona 1 z 28 Następna →

 URocze wycieczki O nas Polityka prywatności Kontakt © 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone

 URocze wycieczki Wycieczki Kraje Kontynenty Użytkownicy Rezerwacje Mój profil Wylogowanie ☀

## Nowy kraj

Nazwa kraju  
Polska

Opis  
Krótki opis kraju...

Powierzchnia (km<sup>2</sup>)  
123456789

Ludność  
35000000

Nazwa kontynentu  
Wybierz kontynent ▾

[Dodaj kraj](#)

 URocze wycieczki O nas Polityka prywatności Kontakt © 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone

## Szczegóły kraju

Poznaj informacje o wybranym kraju.



**Szwecja**

Kraj w Europie Północnej.

**Kontynent:** Europa  
**Powierzchnia:** 450 295 km<sup>2</sup>  
**Ludność:** 10 300 000  
**Liczba wycieczek:** 0

[Powrót do listy](#) [Edytuj](#) [Usuń](#)

[← Poprzedni](#) [Następny →](#)

## Edycja kraju

Nazwa kraju

Opis

Powierzchnia (km<sup>2</sup>)

Ludność

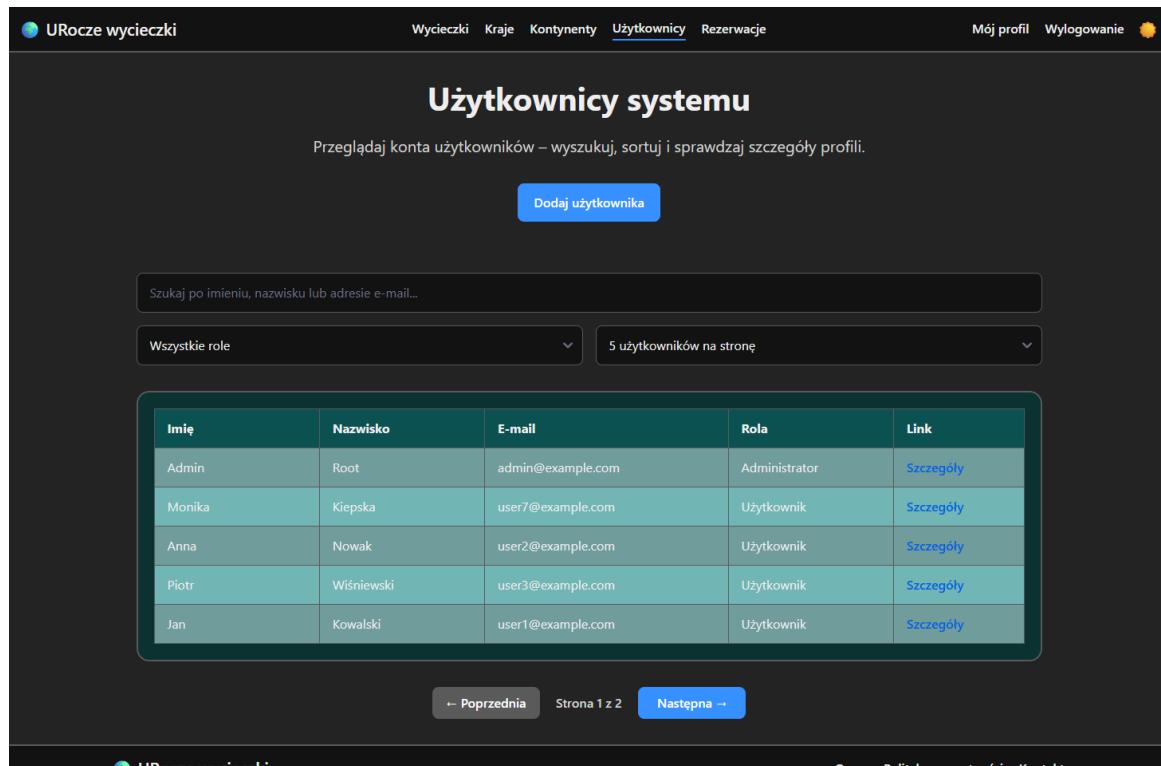
Nazwa kontynentu

Europa

[Zapisz zmiany](#)

## 7.2. Użytkownicy

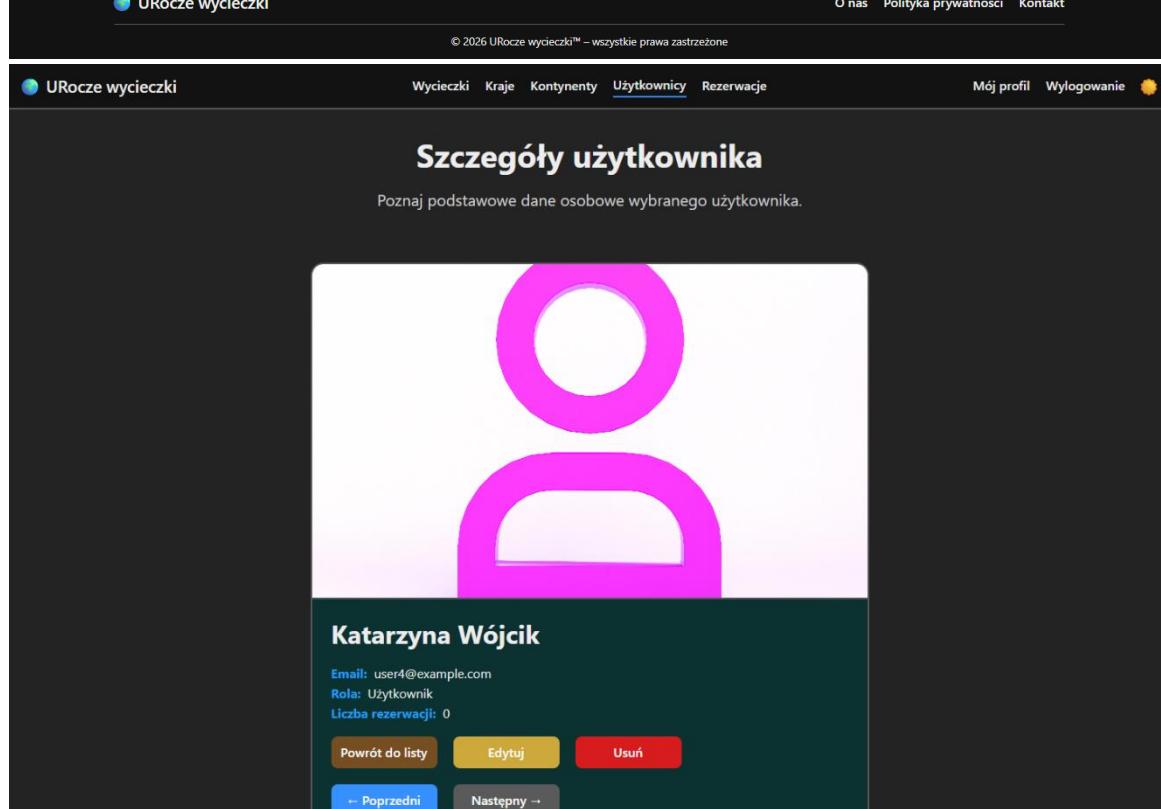
Na tej stronie administrator posiada dostęp do listy wszystkich zarejestrowanych użytkowników w systemie. Widok umożliwia filtrowanie po roli, paginację danych oraz sortowanie i wyszukiwanie. System uniemożliwia usunięcie użytkownika posiadającego rezerwacje.



The screenshot shows the 'Użytkownicy systemu' (System Users) page. At the top, there is a navigation bar with links: 'URocze wycieczki', 'Wycieczki', 'Kraje', 'Kontynenty', 'Użytkownicy' (which is underlined), 'Rezerwacje', 'Mój profil', 'Wylogowanie', and a user icon. Below the navigation is a search bar with placeholder text 'Szukaj po imieniu, nazwisku lub adresie e-mail...' and two dropdown filters: 'Wszystkie role' and '5 użytkowników na stronę'. The main content area displays a table of users:

Imię	Nazwisko	E-mail	Rola	Link
Admin	Root	admin@example.com	Administrator	Szczegóły
Monika	Kiepska	user7@example.com	Użytkownik	Szczegóły
Anna	Nowak	user2@example.com	Użytkownik	Szczegóły
Piotr	Wiśniewski	user3@example.com	Użytkownik	Szczegóły
Jan	Kowalski	user1@example.com	Użytkownik	Szczegóły

At the bottom of the table are navigation buttons: '← Poprzednia', 'Strona 1 z 2', and 'Następna →'.



The screenshot shows the 'Szczegóły użytkownika' (User Details) page for a user named 'Katarzyna Wójcik'. The page features a large pink placeholder image of a person. Below the image, the user's name is displayed in bold. Underneath the name, there is contact information: 'Email: user4@example.com', 'Rola: Użytkownik', and 'Liczba rezerwacji: 0'. At the bottom of the card are three buttons: 'Powrót do listy' (Return to list), 'Edytuj' (Edit), and 'Usuń' (Delete). Navigation buttons '← Poprzedni' and 'Następny →' are also present.

The footer of the page includes links: 'URocze wycieczki', 'O nas', 'Polityka prywatności', 'Kontakt', 'Mój profil', 'Wylogowanie', and a user icon.

## 7.3. Rezerwacje

Administrator widzi pełną listę rezerwacji wraz z informacjami o adresach e-mail użytkowników i nazwami wycieczek. Może wyszukiwać, sortować, filtrować po statusie lub adresie e-mail oraz paginować zbiór danych. Dodatkowo w edycji danej rezerwacji może zmienić status tej rezerwacji.

The screenshot shows the 'Rezerwacje' (Reservations) page of the URocze wycieczki website. At the top, there's a navigation bar with links: 'URocze wycieczki', 'Wycieczki', 'Kraje', 'Kontynenty', 'Użytkownicy', 'Rezerwacje' (which is underlined), 'Mój profil', 'Wylogowanie', and a user icon. Below the navigation is a search bar with placeholder text 'Szukaj po adresie e-mail użytkownika, nazwie wycieczki lub statusie'. Underneath the search bar are three dropdown filters: 'Wszyscy użytkownicy', 'Wszystkie statusy', and '10 rezerwacji na stronę'. The main content area displays a table of 10 reservations:

E-mail użytkownika	Nazwa wycieczki	Rezerwacja	Wycieczka	Status	Link
user1@example.com	Wyprawa do Tokio	09.10.2025	10.10.2025	Zakonczony	Szczegóły
user7@example.com	Trójmiasto	07.10.2025	08.10.2025	Anulowany	Szczegóły
user7@example.com	Zakopane	16.11.2025	21.11.2025	Oczekujący	Szczegóły
user1@example.com	Wyprawa do Pekinu	16.11.2025	02.12.2025	Oczekujący	Szczegóły
user1@example.com	Wakacje w Krakowie	16.11.2025	28.11.2025	Oczekujący	Szczegóły
user3@example.com	Wodny Asuan	15.09.2025	16.09.2025	Zakończony	Szczegóły
user3@example.com	NieOsaka	03.10.2025	04.10.2025	Zakończony	Szczegóły
user7@example.com	Safari w Egipcie	16.11.2025	12.12.2025	Zatwierdzony	Szczegóły
user2@example.com	Lasy Amazonii	21.08.2025	22.08.2025	Anulowany	Szczegóły
user2@example.com	Plażowe Miami	16.11.2025	19.11.2025	Oczekujący	Szczegóły

At the bottom of the page, there are navigation links: '← Poprzednia', 'Strona 1 z 2', 'Następna →', 'O nas', 'Polityka prywatności', and 'Kontakt'. A small note at the bottom center says '© 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone'.

**Szczegóły rezerwacji**

Informacje dotyczące wybranej rezerwacji.

**Rezerwacja nr 4**

**Użytkownik:** user1@example.com  
**Wycieczka:** Wyprawa do Pekinu  
**Data rezerwacji:** 16.11.2025  
**Data wycieczki:** 02.12.2025  
**Status:** Oczekujacy

**Powrót do listy** **Edytuj** **Usuń**

← Poprzedni Następny →

URocze wycieczki | O nas | Polityka prywatności | Kontakt

© 2026 URocze wycieczki™ – wszystkie prawa zastrzeżone

## 8. Endpointy API

W projekcie zastosowano architekturę REST API opartą na frameworku Express.js. Każdy zasób systemu posiada własny zestaw endpointów, które umożliwiają wykonywanie operacji CRUD oraz dodatkowych akcji biznesowych, takich jak uwierzytelnianie czy zarządzanie rezerwacjami. Dokumentacja API została przygotowana z wykorzystaniem Swagger UI oraz standardu OpenAPI 3.0, co pozwala w przejrzysty sposób zaprezentować strukturę żądań i odpowiedzi. Aby wygenerować dokumentację, w plikach routingu zastosowano komentarze zgodne ze specyfikacją Swaggera, które opisują wszystkie endpointy, wymagane parametry, typy odpowiedzi oraz kody błędów.

### 8.1. Konfiguracja i struktura endpointów

Endpointy znajdują się w osobnych plikach w katalogu routes, z podziałem na moduły:

- **users**,
- **continents**,
- **countries**,
- **trips**,
- **reservations**,
- **auth**.

Każdy plik definiuje pełny zestaw operacji dla jednego zasobu oraz przypisuje odpowiednie middleware (np.: verifyToken, isAdmin), a następnie przekazuje logikę do właściwego kontrolera. Endpointy obejmują pełne CRUD:

- **GET** – pobieranie,
- **POST** – dodawanie,
- **PUT** – aktualizowanie,
- **DELETE** – usuwanie.

Przykład fragmentu pliku routes zawierającego endpointy API jest przedstawiony poniżej.

```
13 /**
14  * @swagger
15  * tags:
16  *   name: Reservations
17  *   description: Operacje na rezerwacjach
18 */
19 /**
20  * @swagger
21  * /reservations/count/trip/{trip_id}:
22  *   get:
23  *     summary: Pobieranie liczby rezerwacji wycieczki
24  *     tags: [Reservations]
25  *     parameters:
26  *       - in: path
27  *         name: trip_id
28  *         required: true
29  *       schema:
30  *         type: integer
31  *         description: ID wycieczki
32  *     responses:
33  *       200:
34  *         description: Pobrano liczbę rezerwacji wycieczki.
35  *       400:
36  *         description: Nieprawidłowy ID wycieczki.
37  *       404:
38  *         description: Wycieczka o podanym ID nie istnieje.
39  *       500:
40  *         description: Błąd serwera podczas pobierania liczby rezerwacji wycieczki.
41  */
42 router.get('/count/trip/:trip_id', controller.getReservationsCountByTrip);
43
44 /**
45  * @swagger
46  * /reservations/count/user/{user_id}:
47  *   get:
48  *     summary: Pobieranie liczby rezerwacji użytkownika
49  *     tags: [Reservations]
50  *     security:
51  *       - bearerAuth: []
52  *     parameters:
53  *       - in: path
54  *         name: user_id
55  *         required: true
56  *       schema:
57  *         type: integer
58  *         description: ID użytkownika
59  *     responses:
60  *       200:
61  *         description: Pobrano liczbę rezerwacji użytkownika.
62  *       400:
63  *         description: Nieprawidłowy ID użytkownika.
64  *       401:
65  *         description: Brak tokenu autoryzacyjnego.
66  *       403:
67  *         description: Nieprawidłowy lub wygasły token autoryzacyjny. / Nie posiadasz uprawnień administratora.
68  *       404:
69  *         description: Użytkownik o podanym ID nie istnieje.
70  *       500:
71  *         description: Błąd serwera podczas pobierania liczby rezerwacji użytkownika. / Błąd serwera podczas pobierania użytkownika.
72  */
73 router.get('/count/user/:user_id', verifyToken, isAdmin, controller.getReservationsCountByUser);
74
75 /**
76  * @swagger
77  * /reservations/user/{user_id}:
78  *   get:
79  *     summary: Pobieranie wszystkich rezerwacji użytkownika
80  *     tags: [Reservations]
81  *     security:
82  *       - bearerAuth: []
83  *     parameters:
84  *       - in: path
85  *         name: user_id
86  *         required: true
87  *       schema:
88  *         type: integer
89  *         description: ID użytkownika
90  *     responses:
91  *       200:
92  *         description: Pobrano rezerwacje użytkownika. / Nie znaleziono żadnych rezerwacji użytkownika.
93  *       400:
94  *         description: Nieprawidłowy ID użytkownika.
95  *       401:
96  *         description: Brak tokenu autoryzacyjnego.
97  *       403:
98  *         description: Nieprawidłowy lub wygasły token autoryzacyjny. / Nie posiadasz uprawnień do tych danych.
99  *       404:
100  *         description: Użytkownik o podanym ID nie istnieje.
101  *       500:
102  *         description: Błąd serwera podczas pobierania rezerwacji użytkownika. / Błąd serwera podczas pobierania użytkownika.
103  */
104 router.get('/user/:user_id', verifyToken, isAdminOrSelf, controller.getReservationsByUser);
```

## 8.2. Dokumentacja API w Swagger UI

W projekcie zastosowano dokumentację Swagger UI, dostępną pod adresem /api-docs. Dokumentacja Swagger powstała dzięki komentarzom zapisanym w plikach routingu. Dzięki temu Swagger UI generuje interaktywny interfejs pozwalający:

- Przeglądać wszystkie dostępne endpointy.
- Sprawdzać wymagane parametry.
- Analizować przykładowe odpowiedzi.
- Testować działanie API bez dodatkowych narzędzi (np.: Postman).
- Generować automatyczny podgląd struktury JSON.

Zrzuty ekranu dokumentacji API Swagger UI znajdują się poniżej.

The screenshot displays the Swagger UI interface for a REST API & SPA project titled "Projekt REST API & SPA: Zarządzanie Wycieczkami". The interface includes a navigation bar with the title, version (1.0), and OAS 3.0 badge. Below the bar, there's a search field labeled "REST API do zarządzania wycieczkami" and an "Authorize" button with a lock icon. The main content area is organized into sections: "Auth", "Continents", "Countries", and "Reservations". Each section contains a list of API operations with their methods, URLs, descriptions, and status indicators (green for successful, orange for pending, red for failed). The "Auth" section includes endpoints for registration, login, and token verification. The "Continents" section includes endpoints for listing, creating, reading, updating, and deleting continents. The "Countries" section includes endpoints for listing, creating, reading, updating, and deleting countries. The "Reservations" section includes endpoints for counting trips, getting user reservations, booking trips, and creating new reservations.

Section	Operation	Method	URL	Description	Status
Auth	/auth/register	POST		Rejestracja użytkownika	Green
	/auth/login	POST		Logowanie użytkownika	Green
	/auth/verify	GET		Weryfikacja tokenu JWT	Blue
Continents	/continents	GET		Pobieranie wszystkich kontynentów z obsługą sortowania, wyszukiwania i paginacji	Green
	/continents	POST		Dodawanie nowego kontynentu	Green
	/continents/{id}	GET		Pobieranie jednego kontynentu	Green
	/continents/{id}	PUT		Aktualizacja kontynentu	Orange
	/continents/{id}	DELETE		Usuwanie kontynentu	Red
	Countries	/countries	GET		Pobieranie wszystkich krajów z obsługą filtrowania, sortowania, wyszukiwania i paginacji
/countries		POST		Dodawanie nowego kraju	Green
/countries/{id}		GET		Pobieranie jednego kraju	Green
/countries/{id}		PUT		Aktualizacja kraju	Orange
/countries/{id}		DELETE		Usuwanie kraju	Red
Reservations		/reservations/count/trip/{trip_id}	GET		Pobieranie liczby rezerwacji wycieczki
	/reservations/count/user/{user_id}	GET		Pobieranie liczby rezerwacji użytkownika	Green
	/reservations/user/{user_id}	GET		Pobieranie wszystkich rezerwacji użytkownika	Green
	/reservations/book	POST		Rezerwacja wycieczki przez zalogowanego użytkownika	Green
	/reservations	GET		Pobieranie wszystkich rezerwacji z obsługą filtrowania, sortowania, wyszukiwania i paginacji	Green
	/reservations	POST		Dodawanie nowej rezerwacji	Green

<b>GET</b>	/reservations/{id}	Pobieranie jednej rezerwacji	
<b>PUT</b>	/reservations/{id}	Aktualizacja rezerwacji	
<b>DELETE</b>	/reservations/{id}	Usuwanie rezerwacji	
<b>Trips</b> Operacje na wycieczkach			
<b>GET</b>	/trips	Pobieranie wszystkich wycieczek z obsługą filtrowania, sortowania, wyszukiwania i paginacji	
<b>POST</b>	/trips	Dodawanie nowej wycieczki	
<b>GET</b>	/trips/{id}	Pobieranie jednej wycieczki	
<b>PUT</b>	/trips/{id}	Aktualizacja wycieczki	
<b>DELETE</b>	/trips/{id}	Usuwanie wycieczki	
<b>Users</b> Operacje na użytkownikach			
<b>GET</b>	/users	Pobieranie wszystkich użytkowników z obsługą filtrowania, sortowania, wyszukiwania i paginacji	
<b>POST</b>	/users	Dodawanie nowego użytkownika	
<b>GET</b>	/users/{id}	Pobieranie jednego użytkownika	
<b>PUT</b>	/users/{id}	Aktualizacja użytkownika	
<b>DELETE</b>	/users/{id}	Usuwanie użytkownika	
<b>GET</b>	/users/profile/me	Pobieranie własnych danych zalogowanego użytkownika	
<b>PUT</b>	/users/profile/me	Aktualizacja własnych danych zalogowanego użytkownika	

### 8.3. Zgodność API z 2 poziomem modelu dojrzałości Richardsona

REST API zastosowane w projekcie spełnia drugi poziom modelu dojrzałości Richardsona, ponieważ:

#### 1. Wykorzystuje różne metody HTTP:

- Każda metoda (GET, POST, PUT, DELETE) jest użyta zgodnie ze swoim przeznaczeniem.

#### 2. Posiada osobne endpointy dla zasobów:

- Zasoby są odseparowane (np.: users, trips, reservations) i każdy zasób ma pełny, logiczny zestaw operacji.

#### 3. Korzysta z kodów odpowiedzi HTTP:

- Wykorzystuje różne statusy odpowiedzi:
  - 200 – OK,
  - 201 – Created,
  - 400 – Bad Request,
  - 401 – Unauthorized,
  - 403 – Forbidden,
  - 404 – Not Found,
  - 500 – Server Error.

#### 4. Zasoby zwracane są w formacie JSON:

- Dane są uporządkowane, zrozumiałe i łatwe do przetwarzania przez frontend SPA.

## 9. Logika biznesowa

Logika biznesowa to zbiór zasad, procedur i operacji, które określają, jak aplikacja przetwarza dane i podejmuje decyzje, aby realizować cele biznesowe. Obejmuje reguły, przepływy pracy oraz operacje, które definiują, jak system powinien reagować na różne zdarzenia i przekształcać dane. Logika biznesowa oddziela zasady działania aplikacji od innych jej części, takich jak interfejs użytkownika czy baza danych, zapewniając spójność i łatwość zarządzania.

### 9.1. Realizacja sortowania

Poniżej przedstawiony jest fragment kodu odpowiadający za sortowanie.

```
5   const { sort = 'trips.id', order = 'asc' } = req.query;
6
7   const allowedSortFields = [
8     id: 'trips.id',
9     name: 'trips.name',
10    description: 'trips.description',
11    period: 'trips.period',
12    price: 'trips.price',
13    country_name: 'countries.name',
14    country_id: 'countries.id',
15    continent_name: 'continents.name',
16    continent_id: 'continents.id'
17  ];
18  const allowedOrder = ['ASC', 'DESC'];
19  const safeSort = allowedSortFields[sort] ? allowedSortFields[sort] : 'trips.id';
20  const safeOrder = allowedOrder.includes(order.toUpperCase()) ? order.toUpperCase() : 'ASC';
21
22  const query = `
23    SELECT
24      trips.id,
25      trips.name,
26      trips.description,
27      trips.period,
28      trips.price,
29      countries.id AS country_id,
30      countries.name AS country_name,
31      continents.id AS continent_id,
32      continents.name AS continent_name
33    FROM trips
34    JOIN countries ON trips.country_id = countries.id
35    JOIN continents ON countries.continent_id = continents.id
36    ${whereClause}
37    ORDER BY ${safeSort} ${safeOrder}
38    LIMIT ? OFFSET ?
39  `;
```

Sortowanie odbywa się na podstawie dwóch parametrów: sort oraz order. Najpierw kod sprawdza, czy użytkownik podał poprawne pole do sortowania. Jeśli nie, sortowanie domyślnie odbywa się po ID (w tym kodzie jest to ID wycieczki). Dodatkowo system weryfikuje kolejność sortowania (ASC lub DESC), aby zapobiec wstrzyknięciom SQL. Dzięki temu sortowanie działa w pełni bezpiecznie i pozwala uporządkować dane według np.: nazwy, ceny, kontynentu, kraju.

Przykład sortowania w dokumentacji API Swagger UI przedstawiony jest poniżej.

Name	Description
search string (query)	Wyszukiwanie po nazwie kontynentu, nazwie kraju lub nazwie wycieczki <input type="text" value="search"/>
sort string (query)	Sortowanie po kolumnie (id, name, description, period, price, country_name, country_id, continent_name, continent_id) <input type="text" value="country_name"/>
order string (query)	Kolejność sortowania <input type="text" value="desc"/>
continent_name string (query)	Filtrowanie po nazwie kontynentu <input type="text" value="continent_name"/>
continent_id integer (query)	Filtrowanie po ID kontynentu <input type="text" value="continent_id"/>
country_name string (query)	Filtrowanie po nazwie kraju <input type="text" value="country_name"/>

country\_id  
integer  
(query)

country\_id  
Filtrowanie po ID kraju

page  
integer  
(query)

Numer strony  
1

limit  
integer  
(query)

Liczba wyników na stronę  
5

**Execute** **Clear**

**Responses**

Curl

```
curl -X 'GET' \
'http://localhost:3000/trips?sort=country_name&order=desc&page=1&limit=5' \
-H 'accept: */*'
```

Request URL

[http://localhost:3000/trips?sort=country\\_name&order=desc&page=1&limit=5](http://localhost:3000/trips?sort=country_name&order=desc&page=1&limit=5)

Server response

Code	Details									
200	<b>Response body</b> <pre>{   "message": "Pobrano wycieczki.",   "data": [     {       "id": 33,       "name": "Bogaty Dubaj",       "description": "Luksus, nowoczesna architektura i pustynne safari.",       "period": 10,       "price": 11600,       "country_id": 108,       "country_name": "Zjednoczone Emiraty Arabskie",       "continent_id": 1,       "continent_name": "Azja"     },     {       "id": 35,       "name": "Nowoczesne Abu Zabi",       "description": "Kultura, sztuka i nowoczesne atrakcje.",       "period": 4,       "price": 9700,       "country_id": 108,       "country_name": "Zjednoczone Emiraty Arabskie",       "continent_id": 1,       "continent_name": "Azja"     },     {       "id": 11,       "name": "Wycieczka do Rzymu",       "description": "Historia, kultura i piękno miast.",       "period": 14,       "price": 15000,       "country_id": 109,       "country_name": "Włochy",       "continent_id": 2,       "continent_name": "Europa"     }   ] }</pre> <p><b>Download</b></p> <p><b>Response headers</b></p> <pre>access-control-allow-origin: * connection: keep-alive content-length: 1108 content-type: application/json; charset=utf-8 date: Mon, 17 Nov 2025 10:27:00 GMT etag: "545f0d70d5e5e5f5a5ktw0o" keep-alive: timeout=5 x-powered-by: Express</pre> <p><b>Responses</b></p> <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>Pobrano wycieczki / Nie znaleziono żadnych wycieczek.</td> <td>No links</td> </tr> <tr> <td>500</td> <td>Błąd serwera podczas pobierania wycieczek.</td> <td>No links</td> </tr> </tbody> </table>	Code	Description	Links	200	Pobrano wycieczki / Nie znaleziono żadnych wycieczek.	No links	500	Błąd serwera podczas pobierania wycieczek.	No links
Code	Description	Links								
200	Pobrano wycieczki / Nie znaleziono żadnych wycieczek.	No links								
500	Błąd serwera podczas pobierania wycieczek.	No links								

Przykład w aplikacji zastosowania tej logiki biznesowej został przedstawiony poniżej.

## Nasze wycieczki

Odkryj świat z nami – od egzotycznych plaż po majestatyczne góry. Wybierz kierunek, który rozbudzi Twoją ciekawość!

Szukaj wycieczki, kontynentu lub kraju...

Wszystkie kontynenty ▼ Wszystkie kraje ▼ 5 wycieczek na stronę ▼

Nazwa wycieczki	Kontynent	Kraj	Okres ▼	Cena	Link
Wyprawa do Nowego Jorku	America Północna	Stany Zjednoczone	10 dni	7900 PLN	<a href="#">Szczegóły</a>
Ikoniczne Toronto	America Północna	Kanada	9 dni	4480 PLN	<a href="#">Szczegóły</a>
Mafijna Sycylia	Europa	Włochy	8 dni	5500 PLN	<a href="#">Szczegóły</a>
Nocne Los Angeles	America Północna	Stany Zjednoczone	8 dni	7300 PLN	<a href="#">Szczegóły</a>
Starożytna Ayutthaya	Azja	Tajlandia	8 dni	7500 PLN	<a href="#">Szczegóły</a>

← Poprzednia Strona 1 z 14 Następna →

## 9.2. Realizacja filtrowania

Poniżej przedstawiony jest fragment kodu odpowiadający za filtrowanie.

```
4   const { continent_name, continent_id, country_name, country_id } = req.query;
5
6   if (continent_name) {
7     whereClause += ' AND continents.name = ?';
8     params.push(continent_name);
9   }
10  if (continent_id) {
11    whereClause += ' AND continents.id = ?';
12    params.push(continent_id);
13  }
14  if (country_name) {
15    whereClause += ' AND countries.name = ?';
16    params.push(country_name);
17  }
18  if (country_id) {
19    whereClause += ' AND countries.id = ?';
20    params.push(country_id);
21  }
22
23  const query = `
24   SELECT
25     trips.id,
26     trips.name,
27     trips.description,
28     trips.period,
29     trips.price,
30     countries.id AS country_id,
31     countries.name AS country_name,
32     continents.id AS continent_id,
33     continents.name AS continent_name
34   FROM trips
35   JOIN countries ON trips.country_id = countries.id
36   JOIN continents ON countries.continent_id = continents.id
37   ${whereClause}
38   ORDER BY ${safeSort} ${safeOrder}
39   LIMIT ? OFFSET ?`;
40
```

Filtrowanie w przypadku wycieczek obsługuje filtrację danych według nazw lub identyfikatorów kontynentów oraz krajów. Jeśli użytkownik przekaże w parametrze zapytania np.: continent\_name=Europa, do zapytania SQL zostanie dodany dodatkowy warunek AND. Filtrowanie jest warunkiem nadbudowywanym, czyli można łączyć kilka filtrów naraz np.: continent\_name=Europa AND country\_name=Polska. Dzięki temu użytkownik otrzymuje tylko te wycieczki, które spełniają wybrane kryteria.

Przykład filtrowania w dokumentacji API Swagger UI przedstawiony jest poniżej.

GET /trips Pobieranie wszystkich wycieczek z obsługą filtrowania, sortowania, wyszukiwania i paginacji

Cancel

Parameters

Name	Description
search string (query)	Wyszukiwanie po nazwie kontynentu, nazwie kraju lub nazwie wycieczki <input type="text" value="search"/>
sort string (query)	Sortowanie po kolumnie (id, name, description, period, price, country_name, country_id, continent_name, continent_id) <input type="text" value="id"/>
order string (query)	Kolejność sortowania <input type="text" value="asc"/>
continent_name string (query)	Filtrowanie po nazwie kontynentu <input type="text" value="continent_name"/>
continent_id integer (query)	Filtrowanie po ID kontynentu <input type="text" value="1"/>
country_name string (query)	Filtrowanie po nazwie kraju <input type="text" value="country_name"/>
country_id integer (query)	Filtrowanie po ID kraju <input type="text" value="14"/>
page integer (query)	Numer strony <input type="text" value="1"/>
limit integer (query)	Liczba wyników na stronę <input type="text" value="5"/>

Execute Clear

**Responses**

Curl

```
curl -X 'GET' \
'http://localhost:3000/trips?sort=id&order=asc&continent_id=1&country_id=14&page=1&limit=5' \
-H 'accept: */*'
```

Request URL

```
http://localhost:3000/trips?sort=id&order=asc&continent_id=1&country_id=14&page=1&limit=5
```

Server response

Code	Details	Links
200	<p>Response body</p> <pre>{   "message": "Pobrano wycieczki.",   "data": [     {       "id": 16,       "name": "Wyprawa do Paryża",       "description": "Miasto miłości i sztuki.",       "period": 3200,       "price": 3200,       "country_id": 14,       "country_name": "Francja",       "continent_id": 1,       "continent_name": "Europa"     },     {       "id": 17,       "name": "Muzeum Louvre",       "description": "Największe muzeum sztuki na świecie.",       "period": 3,       "price": 2000,       "country_id": 14,       "country_name": "Francja",       "continent_id": 1,       "continent_name": "Europa"     },     {       "id": 18,       "name": "Dolina Loary",       "description": null     }   ] }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * connection: keep-alive content-length: 810 content-type: application/json; charset=utf-8 date: Mon, 22 May 2023 10:36:48 GMT etag: W/"32a-8627a05nxYKCKs0PVnnDXbaDK" keep-alive: timeout=5 x-powered-by: Express</pre>	
Responses		
Code	Description	Links
200	Pobrano wycieczki. / Nie znaleziono żadnych wycieczek.	No links
500	Błąd serwera podczas pobierania wycieczek.	No links

Przykład w aplikacji zastosowania tej logiki biznesowej został przedstawiony poniżej.

## Nasze wycieczki

Odkryj świat z nami – od egzotycznych plaż po majestatyczne góry. Wybierz kierunek, który rozbudzi Twoją ciekawość!

Australia

Nowa Zelandia

5 wycieczek na stronę

Nazwa wycieczki	Kontynent	Kraj	Okres	Cena	Link
Żeglarskie Auckland	Australia	Nowa Zelandia	8 dni	8000 PLN	<a href="#">Szczegóły</a>
Rotorua	Australia	Nowa Zelandia	5 dni	3840 PLN	<a href="#">Szczegóły</a>
Przygodowe Queenstown	Australia	Nowa Zelandia	6 dni	5230 PLN	<a href="#">Szczegóły</a>

← Poprzednia Strona 1 z 1 Następna →

### 9.3. Realizacja wyszukiwania

Poniżej przedstawiony jest fragment kodu odpowiadający za wyszukiwanie.

```
4  const { search = '' } = req.query;
5
6  const normalizedSearch = normalize(search);
7  const normalizeSQL = `
8      REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
9          REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(
10             'ą', 'a'), 'ć', 'c'), 'ę', 'e'), 'ł', 'l'), 'ń', 'n'), 'ó', 'o'), 'ś', 's'), 'ż', 'z'), 'ź', 'z'),
11             'A', 'A'), 'Ć', 'C'), 'Ę', 'E'), 'Ł', 'L'), 'Ń', 'N'), 'Ó', 'O'), 'Ś', 'S'), 'Ż', 'Z'), 'Ź', 'Z')
12      `;
13
14  let whereClause = `
15      WHERE (
16          LOWER(${normalizeSQL.replace(/col/g, 'continents.name')}) LIKE ?
17          OR LOWER(${normalizeSQL.replace(/col/g, 'countries.name')}) LIKE ?
18          OR LOWER(${normalizeSQL.replace(/col/g, 'trips.name')}) LIKE ?
19      );
20
21  const params = ['${normalizedSearch}', '${normalizedSearch}', '${normalizedSearch}'];
22
23  const query = `
24      SELECT
25          trips.id,
26          trips.name,
27          trips.description,
28          trips.period,
29          trips.price,
30          countries.id AS country_id,
31          countries.name AS country_name,
32          continents.id AS continent_id,
33          continents.name AS continent_name
34      FROM trips
35      JOIN countries ON trips.country_id = countries.id
36      JOIN continents ON countries.continent_id = continents.id
37      ${whereClause}
38      ORDER BY ${safeSort} ${safeOrder}
39      LIMIT ? OFFSET ?
40  `;
```

Powyższy fragment odpowiada za wyszukiwanie po trzech polach: nazwa kontynentu, nazwa kraju oraz nazwa wycieczki. Wyszukiwanie działa w sposób niezależny od polskich znaków, dzięki zastosowaniu funkcji normalizeSQL, która zamienia wszystkie polskie litery na podstawowe odpowiedniki. Następnie ciąg wyszukiwany jest w relacji SQL za pomocą operatora LIKE. W efekcie użytkownik może wpisywać dowolne frazy, a system wyszukuje nawet przy różnicach w pisowni.

Przykład wyszukiwania w dokumentacji API Swagger UI przedstawiony jest poniżej.

The screenshot shows the Swagger UI interface for a 'trips' endpoint. The URL is /trips. The parameters section is expanded, showing the following fields:

Name	Description
search string (query)	Wyszukiwanie po nazwie kontynentu, nazwie kraju lub nazwie wycieczki Value: pol
sort string (query)	Sortowanie po kolumnie (id, name, description, period, price, country_name, country_id, continent_name, continent_id) Value: id
order string (query)	Kolejność sortowania Value: asc
continent_name string (query)	Filtrowanie po nazwie kontynentu Value: continent_name
continent_id integer (query)	Filtrowanie po ID kontynentu Value: continent_id
country_name string (query)	Filtrowanie po nazwie kraju Value: country_name
country_id integer (query)	Filtrowanie po ID kraju Value: country_id
page integer (query)	Numer strony Value: 1
limit integer (query)	Liczba wyników na stronę Value: 5

At the bottom, there are 'Execute' and 'Clear' buttons.

**Responses**

**Curl**

```
curl -X 'GET' \
  'http://localhost:3000/trips?search=pol&sort=idℴ=asc&page=1&limit=5' \
  -H 'accept: */*'
```

**Request URL**

```
http://localhost:3000/trips?search=pol&sort=idℴ=asc&page=1&limit=5
```

**Server response**

Code	Details	Links
200	<p>Response body</p> <pre>{   "message": "Pobrano wycieczki.",   "data": [     {       "id": 1,       "name": "Makajce w Krakowie",       "description": "Zwiedzanie starego miasta.",       "period": 3,       "price": 999,       "country_id": 1,       "country_name": "Polska",       "continent_id": 1,       "continent_name": "Europa"     },     {       "id": 2,       "name": "Trójmiasto",       "description": "Relaks nad Bałtykiem.",       "period": 5,       "price": 1500,       "country_id": 1,       "country_name": "Polska",       "continent_id": 1,       "continent_name": "Europa"     },     {       "id": 14,       "name": "Neapol i Pompeje".     }   ] }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * connection: keep-alive content-length: 1047 content-type: application/json; charset=utf-8 date: Mon, 17 Jul 2023 10:41:32 GMT etag: W/"417-301Kb0MoPGfhwJ0CeTgvIvsQ08" keep-alive: timeout=5 x-powered-by: Express</pre>	
Responses		
Code	Description	Links
200	Pobrano wycieczki. / Nie znaleziono żadnych wycieczek.	No links
500	Błąd serwera podczas pobierania wycieczek.	No links

Przykład w aplikacji zastosowania tej logiki biznesowej został przedstawiony poniżej.

## Nasze wycieczki

Odkryj świat z nami – od egzotycznych plaż po majestatyczne góry. Wybierz kierunek, który rozbudzi Twoją ciekawość!

el
Wszystkie kontynenty
Wszystkie kraje
5 wycieczek na stronę

Nazwa wycieczki	Kontynent	Kraj	Okres	Cena	Link
Nocne Los Angeles	America Północna	Stany Zjednoczone	8 dni	7300 PLN	<a href="#">Szczegóły</a>
Wielka Rafa Koralowa	Australia	Australia	6 dni	5100 PLN	<a href="#">Szczegóły</a>
Melbourne	Australia	Australia	5 dni	3500 PLN	<a href="#">Szczegóły</a>
Wyprawa do Delhi	Azja	Indie	6 dni	3400 PLN	<a href="#">Szczegóły</a>
Żeglarskie Auckland	Australia	Nowa Zelandia	8 dni	8000 PLN	<a href="#">Szczegóły</a>

← Poprzednia
Strona 1 z 2
Następna →

## 9.4. Realizacja paginacji

Poniżej przedstawiony jest fragment kodu odpowiadający za paginację.

```
4   const { page = 1, limit = 5 } = req.query;
5
6   const offset = (page - 1) * limit;
7
8   const query = `
9     SELECT
10       trips.id,
11       trips.name,
12       trips.description,
13       trips.period,
14       trips.price,
15       countries.id AS country_id,
16       countries.name AS country_name,
17       continents.id AS continent_id,
18       continents.name AS continent_name
19     FROM trips
20     JOIN countries ON trips.country_id = countries.id
21     JOIN continents ON countries.continent_id = continents.id
22     ${whereClause}
23     ORDER BY ${safeSort} ${safeOrder}
24     LIMIT ? OFFSET ?
25   `;
26
27   params.push(Number(limit), Number(offset));

```

Paginacja ogranicza liczbę rekordów zwracanych w jednym żądaniu, co znaczco poprawia wydajność aplikacji. Parametr page określa aktualną stronę, natomiast limit określa maksymalną liczbę wycieczek na stronę. Na tej podstawie wyliczany jest offset, czyli liczba rekordów do pominięcia. Dzięki paginacji frontend może płynnie wyświetlać dane w podziale na strony, bez obciążania serwera dużymi zapytaniami.

Przykład paginacji w dokumentacji API Swagger UI przedstawiony jest poniżej.

The screenshot shows the Swagger UI interface for the `/trips` endpoint. The top bar indicates a `GET` method and the URL `/trips`. Below this, the `Parameters` section is expanded, showing various query parameters:

- `search`: string (query) - Wyszukiwanie po nazwie kontynentu, nazwie kraju lub nazwie wycieczki. Value: `search`.
- `sort`: string (query) - Sortowanie po kolumnie (id, name, description, period, price, country\_name, country\_id, continent\_name, continent\_id). Value: `id`.
- `order`: string (query) - Kolejność sortowania. Value: `asc`.
- `continent_name`: string (query) - Filtrowanie po nazwie kontynentu. Value: `continent_name`.
- `continent_id`: integer (query) - Filtrowanie po ID kontynentu. Value: `continent_id`.
- `country_name`: string (query) - Filtrowanie po nazwie kraju. Value: `country_name`.
- `country_id`: integer (query) - Filtrowanie po ID kraju. Value: `country_id`.
- `page`: integer (query) - Numer strony. Value: `5`.
- `limit`: integer (query) - Liczba wyników na stronę. Value: `2`.

At the bottom of the parameters section are two buttons: `Execute` and `Clear`.

Below the parameters section is the `Responses` section, which contains a `Curl` code block and a `Request URL` input field.

```
Curl
curl -X 'GET' \
'http://localhost:3000/trips?sort=id&order=asc&page=5&limit=2' \
-H 'Accept: */*'
```

Request URL: `http://localhost:3000/trips?sort=id&order=asc&page=5&limit=2`

Server response

Code	Details									
200	<p>Response body</p> <pre>{   "message": "Pobrano wycieczki.",   "data": [     {       "id": 9,       "name": "Tokysyczna Hiroshima",       "description": "Historia i odbudowa po II wojnie światowej.",       "period": 4,       "price": 2800,       "country_id": 83,       "country_name": "Japonia",       "continent_id": 2,       "continent_name": "Azja"     },     {       "id": 10,       "name": "Zwiedzanie Kioto",       "description": "Tradycyjna stolica Japonii z licznymi świątyniami.",       "period": 4,       "price": 3500,       "country_id": 83,       "country_name": "Japonia",       "continent_id": 2,       "continent_name": "Azja"     }   ] }</pre> <p><a href="#">Copy</a> <a href="#">Download</a></p> <p>Response headers</p> <pre>access-control-allow-origin: * connection: keep-alive content-length: 457 content-type: application/json; charset=utf-8 date: Wed, 20 Apr 2025 10:47:24 GMT etag: W/"1c9-20e32gqjLxsFRVn/b501033qA" keep-alive: timeout=5 x-powered-by: Express</pre> <p>Responses</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Links</th> </tr> </thead> <tbody> <tr> <td>200</td> <td>Pobrano wycieczki. / Nie znaleziono żadnych wycieczek.</td> <td>No links</td> </tr> <tr> <td>500</td> <td>Błąd serwera podczas pobierania wycieczek.</td> <td>No links</td> </tr> </tbody> </table>	Code	Description	Links	200	Pobrano wycieczki. / Nie znaleziono żadnych wycieczek.	No links	500	Błąd serwera podczas pobierania wycieczek.	No links
Code	Description	Links								
200	Pobrano wycieczki. / Nie znaleziono żadnych wycieczek.	No links								
500	Błąd serwera podczas pobierania wycieczek.	No links								

Przykład w aplikacji zastosowania tej logiki biznesowej został przedstawiony poniżej.

## Nasze wycieczki

Odkryj świat z nami – od egzotycznych plaż po majestatyczne góry. Wybierz kierunek, który rozbudzi Twoją ciekawość!

Nazwa wycieczki	Kontynent	Kraj	Okres	Cena	Link
Podróż do Singapuru	Azja	Singapur	5 dni	3300 PLN	<a href="#">Szczegóły</a>
Marina Bay Sands	Azja	Singapur	3 dni	2000 PLN	<a href="#">Szczegóły</a>
Zakopane	Europa	Polska	6 dni	1400 PLN	<a href="#">Szczegóły</a>
Sentosa	Azja	Singapur	4 dni	2500 PLN	<a href="#">Szczegóły</a>
Ogród Botaniczny	Azja	Singapur	2 dni	1500 PLN	<a href="#">Szczegóły</a>
Zwiedzanie Vancouver	Ameryka Północna	Kanada	6 dni	35600 PLN	<a href="#">Szczegóły</a>
Wyprawa do Delhi	Azja	Indie	6 dni	3400 PLN	<a href="#">Szczegóły</a>
Francuskie Montreal	Ameryka Północna	Kanada	4 dni	3000 PLN	<a href="#">Szczegóły</a>
Europejskie Quebec City	Ameryka Północna	Kanada	4 dni	2800 PLN	<a href="#">Szczegóły</a>
Piękne Buenos Aires	Ameryka Południowa	Argentyna	6 dni	3670 PLN	<a href="#">Szczegóły</a>

[← Poprzednia](#)
Strona 5 z 7
[Następna →](#)

## 10. Uruchomienie aplikacji

### 10.1. Wymagania

1. **Visual Studio Code:** Jest to wszechstronne, lekkie, potężne i popularne środowisko programistyczne, które jest szczególnie cenione za swoją szybkość, łatwość użycia oraz możliwość dostosowania do indywidualnych potrzeb programisty. VSC wspiera wiele języków programowania i technologii, oferując bogaty zestaw funkcji, takich jak podświetlanie składni, automatyczne uzupełnianie kodu czy debugowanie. Program jest dostępny do pobrania z linku <https://code.visualstudio.com/download>.
2. **Node.js:** Jest to środowisko uruchomieniowe JavaScript wykorzystywanym przez backend aplikacji. Umożliwia instalowanie zależności i uruchamianie serwera Express. Aby rozpocząć korzystanie z Node.js, należy pobrać instalator ze strony internetowej <https://nodejs.org/en/download>.

### 10.2. Kroki przy pierwszym uruchomieniu

1. Pobierz projekt na komputer, otwórz Visual Studio Code i przejdź do folderu projektu.
2. W terminalu przejdź do katalogu /backend i wpisz: npm install.
3. Backend automatycznie tworzy bazę SQLite przy pierwszym uruchomieniu wraz z danymi startowymi. Nie jest wymagane tworzenie bazy ręcznie.
4. Po zainstalowaniu wszystkich zależności w tym samym folderze wpisz: node app.js.
5. Backend uruchomi się pod adresem <http://localhost:3000> a dokumentacja API Swagger UI będzie dostępna pod adresem <http://localhost:3000/api-docs>.
6. Następnie w terminalu przejdź do katalogu /frontend i wpisz: npm install.
7. Polecenie zainstaluje Vue, Vite, Pinia, Vue Router oraz pozostałe moduły SPA.
8. Po zainstalowaniu pakietów w tym samym folderze wpisz: npm run dev.
9. Frontend domyślnie powinien uruchomić się pod adresem <http://localhost:5173>. Aplikacja automatycznie połączy się z backendem poprzez REST API.

### 10.3. Kroki przy kolejnym uruchomieniu

1. Otwórz program VSC i przejdź do folderu projektu.
2. W terminalu przejdź do katalogu /backend i wpisz: node app.js.
3. Następnie w terminalu przejdź do katalogu /frontend i wpisz: npm run dev.
4. Aplikacja będzie dostępna pod adresem <http://localhost:5173>. Backend działać będzie pod adresem <http://localhost:3000> wraz z dokumentacją API dostępną pod adresem <http://localhost:3000/api-docs>.