

AI1	Dokumentacja projektu
Autor	Rafał Liszcz, 125140
Kierunek, rok	Informatyka, II rok, st. stacjonarne (3,5-l)
Temat projektu	Informator o rozkładzie autobusów

Spis treści

Wstęp	3
Narzędzia i technologie	3
Baza danych.....	4
GUI.....	5
Uruchomienie aplikacji	7
Funkcjonalności aplikacji	7
Wybrany proces/logika biznesowa:.....	12
Walidacja danych:	15
Podsumowanie:.....	24

Wstęp

Aplikacja „Rozkład autobusowy” ma na celu umożliwienie użytkownikom wyszukiwania oraz zarządzania rozkładami autobusów. Użytkownicy mogą wyszukiwać przejazdy według miast wyjazdu i przyjazdu, daty oraz godziny wyjazdu. Administratorzy mają dodatkowo możliwość dodawania, edytowania oraz usuwania rozkładów autobusowych.

Narzędzia i technologie

W projekcie wykorzystano następujące technologie:

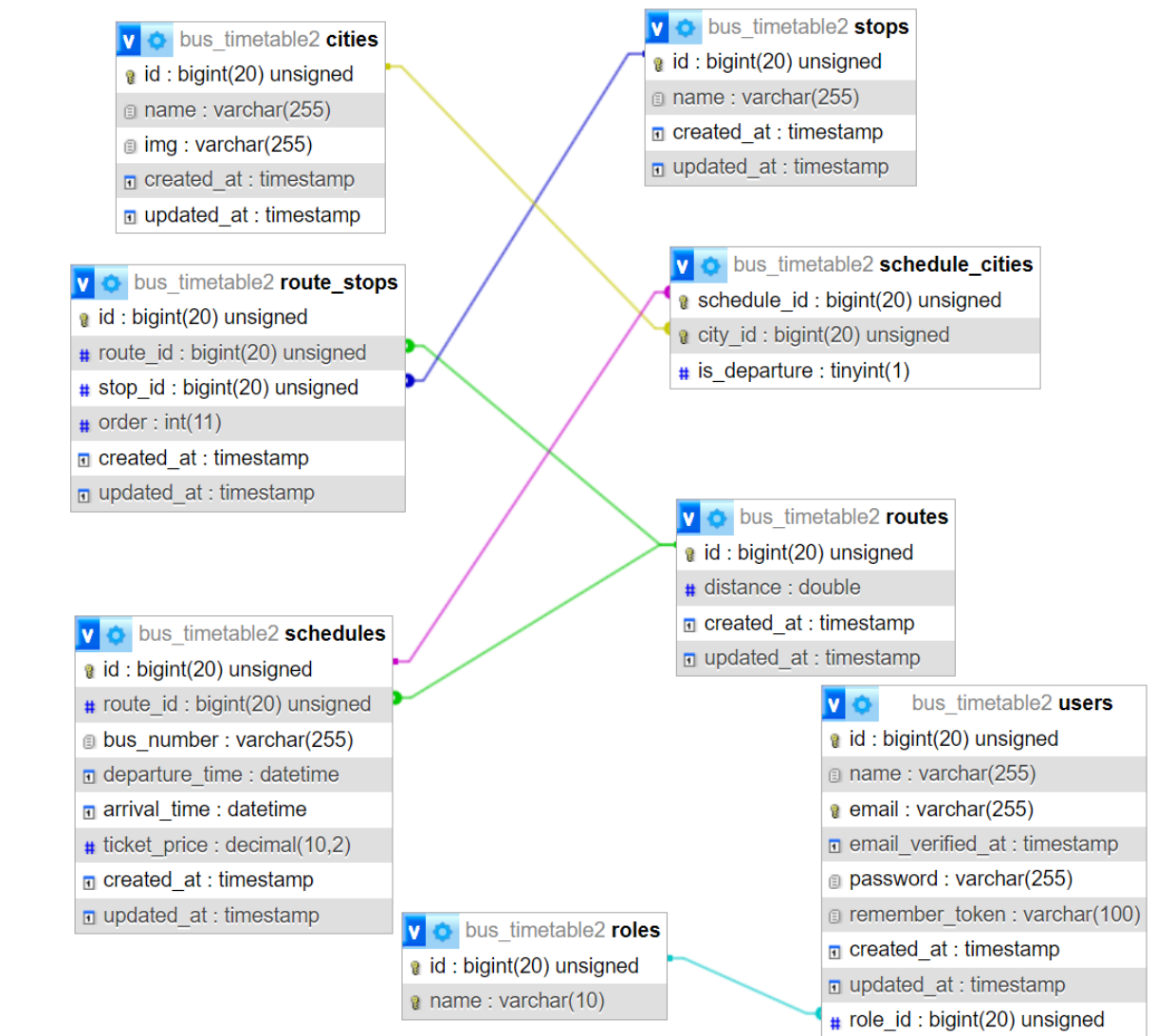
- **PHP (wersja 8.X)** – język programowania używany do tworzenia logiki aplikacji.
- **Laravel (wersja 11.x)** – framework PHP ułatwiający tworzenie aplikacji webowych.
- **Blade** – silnik szablonów wykorzystywany w Laravelu.
- **MySQL** – system zarządzania bazą danych używany do przechowywania danych aplikacji.
- **Bootstrap** – framework CSS używany do stylizacji interfejsu użytkownika.
- **JavaScript** – język programowania używany do dynamicznej interakcji na stronie.

Wszystkie wymienione technologie są dostępne bezpłatnie. Dokumentacja i miejsca pobierania:

- [PHP](#)
- [Laravel](#)
- [MySQL](#)
- [Bootstrap](#)
- [JavaScript](#)

Baza danych

Schemat ERD:



Opis zawartości bazy danych:

- **Users** - tabela przechowująca informacje o użytkownikach (id, name, email, password, role_id).
- **Cities** - tabela przechowująca informacje o miastach (id, name, img, zip_code).
- **Schedules** - tabela przechowująca informacje o rozkładach jazdy (id, departure_time, arrival_time, ticket_price, route_id, bus_number).
- **Routes** - tabela przechowująca informacje o trasach (id, distance).
- **Stops** - tabela przechowująca informacje o przystankach (id, name).

Opis powiązań pomiędzy tabelami:

Jeden użytkownik może być administratorem lub zwykłym użytkownikiem.

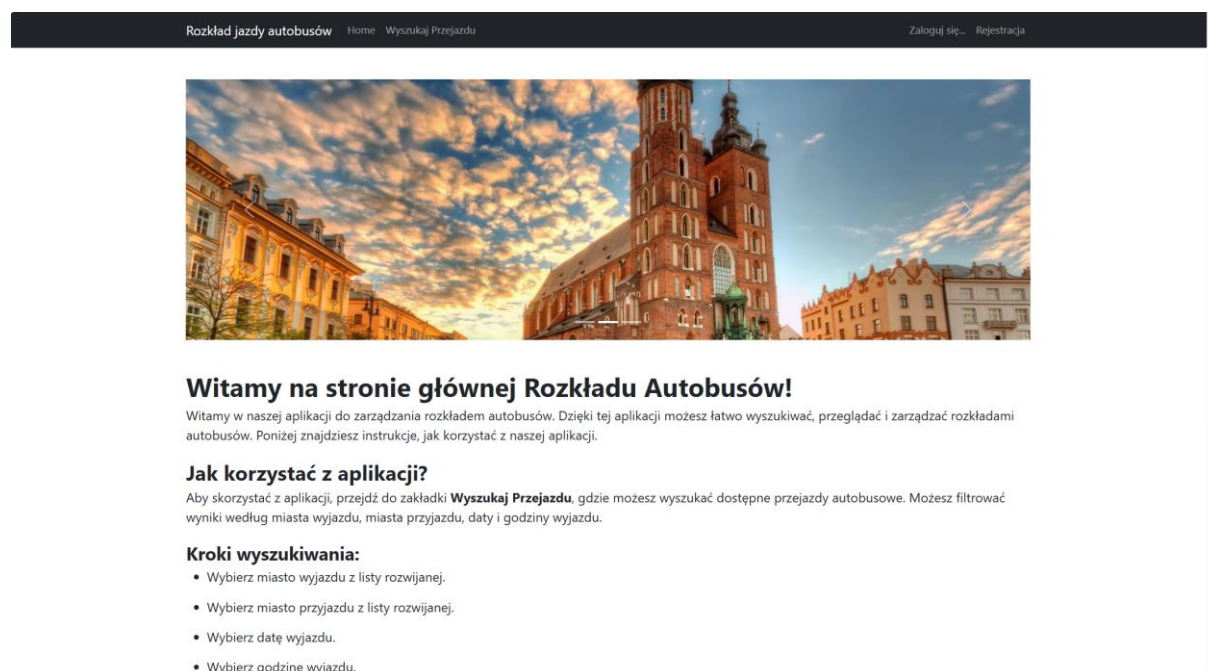
Jeden rozkład jazdy należy do jednej trasy.

Trasa może mieć wiele przystanków.

Miasta są powiązane z rozkładami jazdy przez tabele pośrednie.

GUI

Widok 1: Strona główna



Widok 2: Wyszukiwanie przejazdów

Rozkład jazdy autobusów

Home

Wyszukaj Przejazd

Zaloguj się...

Rejestracja


Wyszukaj przejazd

Wybierz miasto

Wybierz miasto

mm/dd/yyyy

Wyszukaj



A123

Trasa: Warszawa → Kraków


Przystanki: Przystanek 1, Przystanek 2, Przystanek 3


Odległość: 300 km

Odjazd: 2024-06-01 08:00

Przyjazd: 2024-06-01 12:00

Cena biletu: 50.00 zł





B423

Trasa: Wrocław → Poznań


Przystanki: Przystanek 1, Przystanek 4, Przystanek 5


Odległość: 151 km

Odjazd: 2024-06-02 09:00

Przyjazd: 2024-06-02 11:30

Cena biletu: 20.00 zł





C789

Trasa: Gdańsk → Łódź


Przystanki: Przystanek 1, Przystanek 3, Przystanek 5

Odległość: 220 km

Odjazd: 2024-06-03 10:00

Przyjazd: 2024-06-03 14:00

Cena biletu: 45.00 zł



Widok 3: Dodawanie nowego przejazdu

Rozkład jazdy autobusów

Home

Wyszukaj Przejazdu

Dodaj Przejazd

Jan, wyloguj się...

Dodaj Przejazd

Numer Autobusu

Dystans (km)

Miasto Wyjazdu

Choose File

No file chosen

Miasto Przyjazdu

Choose File

No file chosen

Godzina Odjazdu

mm/dd/yyyy --:--

Godzina Przyjazdu

mm/dd/yyyy --:--

Cena

Przystanki (Przystanki muszą być oddzielone przecinkiem)

Dodaj Przejazd

© Rozkład autobusowy – 2024

6

Uruchomienie aplikacji

Kroki konfiguracyjne:

Uruchomić XAMPP oraz Włączyć Apache oraz MySQL.

Otworzyć Projekt w VSCode.

W przypadku posiadania innych ustawień niż domyślne (np. połączenia z bazą), wykonać ich zmianę w `.env.example` oraz `start.bat` lub `start.sh`.

W Folderze Projektu Uruchomić skrypt `start.bat` (Windows, 2x kliknięciem) lub `start.sh` (inne systemy, przez polecenie `bash start.sh`).

Otworzyć terminal `cmd` (Command Prompt) w VSCode.

Uruchom serwer lokalny za pomocą komend `php artisan serve`.

Funkcjonalności aplikacji

Logowanie i rejestracja

Logowanie: Formularz logowania z walidacją danych (email, hasło). Przykładowe loginy i hasła:

Email: `Jan@email.com`, Hasło: `1234` ← Admin

Email: `marta@email.com`, Hasło: `1234` ← Użytkownik

Rozkład jazdy autobusów Home Wyszukaj Przejazdu Zaloguj się... Rejestracja

Zaloguj się

Email
jan@email.com

Hasło

Wyślij

© Rozkład autobusowy - 2024

Rejestracja: Formularz rejestracji z walidacją danych (nazwa użytkownika, email, hasło, potwierdzenie hasła).

Rozkład jazdy autobusów

Home

Wyszukaj Przejazdu

Zaloguj się...

Rejestracja

Zarejestruj się

Nazwa użytkownika

rafal

Email

email@email.com




Hasło

Potwierdź hasło

1234

Zarejestruj się

© Rozkład autobusowy – 2024



CRUD przeprowadzany przez administratora

Dodawanie nowego przejazdu: Administrator może dodawać nowe przejazdy poprzez formularz.

Rozkład jazdy autobusów

Home

Wyszukaj Przejazdu

Dodaj Przejazd

Jan, wyloguj się...

Dodaj Przejazd

Numer Autobusu

RZE123

Dystans (km)

123

Miasto Wyjazdu

Rzeszów

Choose File

Rzeszów-1024x493.png

Miasto Przyjazdu

Głogów Małopolski

Choose File

Głogów_Małopolski_(Rynek)_-_ratusz.png

Godzina Odejazdu

05/28/2024 12:08 AM

Godzina Przyjazdu

05/28/2024 12:14 AM

Cena




12

Przystanki (Przystanki muszą być oddzielone przecinkiem)



Głogów,rudna mała,rudna wielka,Rzeszów

Dodaj Przejazd

© Rozkład autobusowy – 2024





Usuwanie Przejazdu: Administrator może usuwać przejazdy poprzez Kliknięcie w przycisk Delete.





A123
Trasa: Warszawa → Kraków
Przystanki: Przystanek 1, Przystanek 2, Przystanek 3
Odległość: 300 km
Odjazd: 2024-06-01 08:00
Przyjazd: 2024-06-01 12:00
Cena biletu: 50.00 zł

[Edit](#) [Delete](#)



B423
Trasa: Wrocław → Poznań
Przystanki: Przystanek 1, Przystanek 4, Przystanek 5
Odległość: 151 km
Odjazd: 2024-06-02 09:00
Przyjazd: 2024-06-02 11:30
Cena biletu: 20.00 zł



[Edit](#) [Delete](#)



C789
Trasa: Gdańsk → Łódź
Przystanki: Przystanek 1, Przystanek 3, Przystanek 5
Odległość: 220 km
Odjazd: 2024-06-03 10:00
Przyjazd: 2024-06-03 14:00
Cena biletu: 45.00 zł


[Edit](#) [Delete](#)

< 1 2 >




B423
Trasa: Wrocław → Poznań
Przystanki: Przystanek 1, Przystanek 4, Przystanek 5
Odległość: 151 km
Odjazd: 2024-06-02 09:00
Przyjazd: 2024-06-02 11:30
Cena biletu: 20.00 zł

[Edit](#) [Delete](#)



C789
Trasa: Gdańsk → Łódź
Przystanki: Przystanek 1, Przystanek 3, Przystanek 5
Odległość: 220 km
Odjazd: 2024-06-03 10:00
Przyjazd: 2024-06-03 14:00
Cena biletu: 45.00 zł

[Edit](#) [Delete](#)





D012
Trasa: Lublin → Katowice
Przystanki: Przystanek 1, Przystanek 2, Przystanek 4
Odległość: 350 km
Odjazd: 2024-06-04 11:00
Przyjazd: 2024-06-04 16:00
Cena biletu: 60.00 zł

[Edit](#) [Delete](#)

< 1 2 >

Edycja Przejazdu: Administrator może edytować przejazdy poprzez formularz.



B423

Trasa: Wrocław → Poznań

Przystanki: Przystanek 1, Przystanek 4, Przystanek 5

Odległość: 151 km

Odjazd: 2024-06-02 09:00

Przyjazd: 2024-06-02 11:30

Cena biletu: 20.00 zł

EditDelete

Rozkład jazdy autobusówHomeWyszukaj PrzejazdDodaj PrzejazdJan, wyloguj się...

Edytuj Przejazd

Numer Autobusu

B423

Dystans (km)

151

Miasto Wyjazdu

Wrocław

Choose FileNo file chosen

Miasto Przyjazdu

Poznań

Choose FileNo file chosen

Godzina Odjazdu

06/02/2024 09:00 AM

Godzina Przyjazdu

06/02/2024 11:30 AM

Cena

20.00



Przystanki (Przystanki muszą być oddzielone przecinkiem)

Przystanek 1, Przystanek 4, Przystanek 5

Zaktualizuj Przejazd

© Rozkład autobusowy – 2024





B400

Trasa: Wrocław → Poznań

Przystanki: Przystanek 1, Przystanek 4, Przystanek 5

Odległość: 200 km

Odjazd: 2024-06-02 09:00

Przyjazd: 2024-06-02 11:30

Cena biletu: 10.00 zł

EditDelete

Przeglądanie ogólnodostępnych zasobów

Filtrowanie zasobów: Wszyscy mogą filtrować przejazdy według miasta wyjazdu, miasta przyjazdu, daty i godziny wyjazdu.

Rozkład jazdy autobusówHomeWyszukaj Przejazd

Zaloguj się...Rejestracja

Wyszukaj przejazd


Wybierz miasto

Wybierz miasto

mm/dd/yyyy

...

Wyszukaj



A123

Trasa: Warszawa → Kraków


Przystanki: Przystanek 1, Przystanek 2, Przystanek 3

Odległość: 300 km

Odjazd: 2024-06-01 08:00

Przyjazd: 2024-06-01 12:00

Cena biletu: 50.00 zł



B423

Trasa: Wrocław → Poznań


Przystanki: Przystanek 1, Przystanek 4, Przystanek 5

Odległość: 151 km

Odjazd: 2024-06-02 09:00

Przyjazd: 2024-06-02 11:30

Cena biletu: 20.00 zł



C789

Trasa: Gdańsk → Łódź

Przystanki: Przystanek 1, Przystanek 3, Przystanek 5

Odległość: 220 km

Odjazd: 2024-06-03 10:00

Przyjazd: 2024-06-03 14:00


Cena biletu: 45.00 zł


1


?

+

© Rozkład autobusowy - 2024







Rozkład jazdy autobusówHomeWyszukaj Przejazdu

Dodaj PrzejazdJan, wyloguj się...

Wyszukaj przejazd


Rzeszów

Wybierz miasto

mm/dd/yyyy

...

Wyszukaj



RZE123

Trasa: Rzeszów → Głogów Małopolski

Przystanki: Głogów,rudna mała,rudna wielka,Rzeszów

Odległość: 123 km

Odjazd: 2024-05-27 19:24

Przyjazd: 2024-05-27 23:28

Cena biletu: 12.00 zł

Edytuj

Delete

11

Wybrany proces/logika biznesowa:

Wyszukiwanie przejazdów:

Proces wyszukiwania przejazdów polega na filtrowaniu dostępnych rozkładów jazdy na podstawie kryteriów wprowadzonych przez użytkownika. Kryteria te mogą obejmować miasto wyjazdu, miasto przyjazdu, datę oraz godzinę wyjazdu. Poniżej znajduje się implementacja tego procesu:

Pobieranie danych z ządania:

```
$departureCity = $request->input('departure_city');  
$arrivalCity = $request->input('arrival_city');  
$departureDate = $request->input('departure_date');  
$departureTime = $request->input('departure_time');
```

Funkcja pobiera dane z obiektu Request, które zostały przekazane przez użytkownika za pomocą formularza wyszukiwania. Te dane to:

- **departure_city**: Miasto wyjazdu.
- **arrival_city**: Miasto przyjazdu.
- **departure_date**: Data wyjazdu.
- **departure_time**: Godzina wyjazdu.

Inicjalizacja zapytania do bazy danych:

```
$query = Schedule::query();
```

Tworzy początkowe zapytanie do bazy danych na modelu Schedule, które później będzie modyfikowane w zależności od podanych kryteriów.

Dodawanie warunków do zapytania:

Miasto wyjazdu:

```
if ($departureCity) {  
    $query->whereHas('departureCities', function ($query) use  
($departureCity) {  
        $query->where('city_id', $departureCity);  
    });  
}
```

Jeśli użytkownik podał miasto wyjazdu, zapytanie zostanie rozszerzone o warunek, który sprawdza, czy departureCities (relacja w modelu Schedule) zawiera podane city_id.

Miasto przyjazdu:

```
if ($arrivalCity) {  
    $query->whereHas('arrivalCities', function ($query) use ($arrivalCity)  
    {  
        $query->where('city_id', $arrivalCity);  
    });  
}
```

Jeśli użytkownik podał miasto przyjazdu, zapytanie zostanie rozszerzone o warunek, który sprawdza, czy arrivalCities (relacja w modelu Schedule) zawiera podane city_id.

Data wyjazdu:

```
if ($departureDate) {  
    $query->whereDate('departure_time', $departureDate);  
}
```

Jeśli użytkownik podał datę wyjazdu, zapytanie zostanie rozszerzone o warunek, który sprawdza, czy departure_time jest równy podanej dacie.

Godzina wyjazdu:

```
if ($departureTime) {  
    $query->whereTime('departure_time', '>=', $departureTime);  
}
```

Jeśli użytkownik podał godzinę wyjazdu, zapytanie zostanie rozszerzone o warunek, który sprawdza, czy departure_time jest większy lub równy podanej godzinie.

Sprawdzanie, czy żadne kryteria nie zostały podane:

```
if (!$departureCity && !$arrivalCity && !$departureDate && !$departureTime) {  
    $schedules = collect();  
} else {  
    $schedules = $query->with('departureCities', 'arrivalCities',  
'route')->get();  
}
```

Jeśli żadne kryteria nie zostały podane, funkcja zwraca pustą kolekcję. W przeciwnym razie wykonuje zapytanie do bazy danych, pobierając wyniki wraz z powiązаныmi miastami wyjazdu, przyjazdu oraz trasą.

Paginacja wyników:

```
$schedules = $query->with('departureCities', 'arrivalCities', 'route')->paginate(3)->appends([
    'departure_city' => $departureCity,
    'arrival_city' => $arrivalCity,
    'departure_date' => $departureDate,
    'departure_time' => $departureTime,
]);
```

Wyniki są paginowane, aby na jednej stronie wyświetlać tylko trzy rekordy. Funkcja `appends` dodaje parametry wyszukiwania do paginacji, aby zachować je w URL podczas przeglądania kolejnych stron wyników.

Pobieranie wszystkich miast:

```
$cities = City::all();
```

Pobiera wszystkie miasta z bazy danych, aby mogły być wyświetlone w formularzu wyszukiwania.

Zwracanie widoku z wynikami wyszukiwania:

```
return view('index', compact('schedules', 'cities', 'departureCity',
    'arrivalCity', 'departureDate', 'departureTime'));
```

Funkcja zwraca widok `index` z przekazanymi danymi:

`schedules`: Wyniki wyszukiwania.

`cities`: Lista wszystkich miast.

`departureCity, arrivalCity, departureDate, departureTime`: Kryteria wyszukiwania wprowadzone przez użytkownika.

Walidacja danych:

Walidacja danych to kluczowy element każdej aplikacji, który zapewnia, że wprowadzone przez użytkowników informacje są poprawne i zgodne z wymaganiami systemu. W projekcie „Rozkład autobusowy” zastosowano walidację w różnych miejscach, aby zapewnić integralność danych i poprawność działania aplikacji. Poniżej przedstawiono szczegółowy opis walidacji danych dla różnych funkcjonalności.

1. Rejestracja użytkownika

Walidacja danych podczas rejestracji użytkownika odbywa się w metodzie `register` w kontrolerze `AuthController`. Obejmuje ona sprawdzenie, czy nazwa użytkownika, email i hasło spełniają określone wymagania.

```
public function register(Request $request)
{
    $validatedData = $request->validate([
        'name' => 'required|max:255',
        'email' => 'required|email|unique:users',
        'password' => 'required|confirmed',
    ]);

    $user = User::create([
        'name' => $validatedData['name'],
        'email' => $validatedData['email'],
        'password' => Hash::make($validatedData['password']),
        'role_id' => 2,
    ]);

    Auth::login($user);

    return redirect()->route('search');
}
```

name: Wymagane, maksymalnie 255 znaków.

email: Wymagane, musi być unikalny w tabeli users, musi mieć format emaila.

password: Wymagane, musi być potwierdzone (pole password_confirmation).

```

<form method="POST" action="{{ route('register') }}" class="needs-validation" novalidate>
    @csrf
    <div class="form-group mb-2">
        <label for="name" class="form-label">Nazwa użytkownika</label>
        <input id="name" name="name" type="text" class="form-control @if ($errors->first('name')) is-
invalid @endif" value="{{ old('name') }}" required>
        <div class="invalid-feedback">Nazwa użytkownika jest wymagana!</div>
    </div>
    <div class="form-group mb-2">
        <label for="email" class="form-label">Email</label>
        <input id="email" name="email" type="email" class="form-control @if ($errors->first('email'))
is-invalid @endif" value="{{ old('email') }}" required>
        <div class="invalid-feedback">Nieprawidłowy email!</div>
    </div>
    <div class="form-group mb-2">
        <label for="password" class="form-label">Hasło</label>
        <input id="password" name="password" type="password" class="form-control @if ($errors->
first('password')) is-invalid @endif" required>
        <div class="invalid-feedback">Hasło jest wymagane!</div>
    </div>
    <div class="form-group mb-2">
        <label for="password_confirmation" class="form-label">Potwierdź hasło</label>
        <input id="password_confirmation" name="password_confirmation" type="password" class="form-
control @if ($errors->first('password_confirmation')) is-invalid @endif" required>
        <div class="invalid-feedback">Potwierdzenie hasła jest wymagane!</div>
    </div>
    <div class="text-center mt-4 mb-4">
        <input class="btn btn-primary" type="submit" value="Zarejestruj się">
    </div>
</form>

```

Walidacja danych po stronie klienta i servera:

Zarejestruj się

- Pole nazwa jest wymagane.
- Pole adres e-mail jest wymagane.
- Pole hasło jest wymagane.

Nazwa użytkownika

Nazwa użytkownika jest wymagana!

Email

Nieprawidłowy email!

Hasło

Hasło jest wymagane!

Potwierdź hasło

Zarejestruj się

2. Logowanie użytkownika

Walidacja danych podczas logowania użytkownika odbywa się w metodzie `authenticate` w kontrolerze `AuthController`. Obejmuje ona sprawdzenie, czy email i hasło są podane.

```
public function authenticate(Request $request)
{
    $credentials = $request->validate([
        'email' => ['required', 'email'],
        'password' => ['required'],
    ]);

    if (Auth::attempt($credentials)) {
        $request->session()->regenerate();
        return redirect()->route('search');
    }

    return back()->withErrors([
        'email' => 'The provided credentials do not match our records.',
    ])->onlyInput('email');
}
```

email: Wymagane, musi mieć format email.

password: Wymagane.

```
<form method="POST" action="{{ route('login.authenticate') }}" class="needs-validation" novalidate>
    @csrf
    <div class="form-group mb-2">
        <label for="email" class="form-label">Email</label>
        <input id="email" name="email" type="text" class="form-control @if ($errors->first('email'))
is-invalid @endif" value="{{ old('email') }}">
        <div class="invalid-feedback">Nieprawidłowy email!</div>
    </div>
    <div class="form-group mb-2">
        <label for="password" class="form-label">Hasło</label>
        <input id="password" name="password" type="password" class="form-control @if ($errors->
first('password')) is-invalid @endif">
        <div class="invalid-feedback">Nieprawidłowe hasło!</div>
    </div>
    <div class="text-center mt-4 mb-4">
        <input class="btn btn-primary" type="submit" value="Wyślij">
    </div>
</form>
```

Walidacja po stronie klienta i servera:

Zaloguj się

- Pole adres e-mail jest wymagane.
- Pole hasło jest wymagane.

Email

Nieprawidłowy email!

Hasło

Nieprawidłowe hasło!

Wyślij

3. Dodawanie nowego przejazdu

Walidacja danych podczas dodawania nowego przejazdu odbywa się w metodzie `store` kontrolera `ScheduleController` i wykorzystuje klasę `StoreScheduleRequest`.

```
public function rules(): array
{
    return [
        'bus_number' => 'required|string|max:255',
        'distance' => 'required|numeric|min:0',
        'departure_city' => 'required|string|max:255',
        'arrival_city' => 'required|string|max:255',
        'departure_time' => 'required|date|date_format:Y-m-d\TH:i',
        'arrival_time' =>
            'required|date|after:departure_time|date_format:Y-m-d\TH:i',
        'ticket_price' => 'required|numeric|min:0',
        'departure_city_img' =>
            'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        'arrival_city_img' =>
            'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        'stops' => 'array',
        'stops.*' => 'string|max:255',
    ];
}
```

bus_number: Wymagane, tekst, maksymalnie 255 znaków.

distance: Wymagane, liczba, minimum 0.

departure_city: Wymagane, tekst, maksymalnie 255 znaków.

arrival_city: Wymagane, tekst, maksymalnie 255 znaków.

departure_time: Wymagane, data w formacie Y-m-d\TH:i.

arrival_time: Wymagane, data po departure_time, format Y-m-d\TH:i.

ticket_price: Wymagane, liczba, minimum 0.

departure_city_img: Opcjonalne, obrazek w formatach jpeg, png, jpg, gif, maksymalnie 2048 KB.

arrival_city_img: Opcjonalne, obrazek w formatach jpeg, png, jpg, gif, maksymalnie 2048 KB.

stops: Tablica, elementy muszą być tekstem, maksymalnie 255 znaków.

```
<div class="form-group mb-2">
  <label for="bus_number" class="form-label">Numer Autobusu</label>
  <input type="text" name="bus_number" id="bus_number" class="form-control @if ($errors-
>first('bus_number')) is-invalid @endif required maxlength="255">
  <div class="invalid-feedback">Numer autobusu jest wymagany i nie może przekraczać 255 znaków.</div>
</div>
<div class="form-group mb-2">
  <label for="distance" class="form-label">Dystans (km)</label>
  <input type="number" step="0.01" name="distance" id="distance" class="form-control @if ($errors-
>first('distance')) is-invalid @endif required min="0">
  <div class="invalid-feedback">Dystans jest wymagany i musi być liczbą nieujemną.</div>
</div>
<div class="form-group mb-2">
  <label for="departure_city" class="form-label">Miasto Wyjazdu</label>
  <input type="text" name="departure_city" id="departure_city" class="form-control @if ($errors-
>first('departure_city')) is-invalid @endif required maxlength="255">
  <div class="invalid-feedback">Miasto wyjazdu jest wymagane i nie może przekraczać 255 znaków.</div>
  <input type="file" name="departure_city_img" class="form-control mt-2">
</div>
<div class="form-group mb-2">
  <label for="arrival_city" class="form-label">Miasto Przyjazdu</label>
  <input type="text" name="arrival_city" id="arrival_city" class="form-control @if ($errors-
>first('arrival_city')) is-invalid @endif required maxlength="255">
  <div class="invalid-feedback">Miasto przyjazdu jest wymagane i nie może przekraczać 255 znaków.</div>
  <input type="file" name="arrival_city_img" class="form-control mt-2">
</div>
<div class="form-group mb-2">
  <label for="departure_time" class="form-label">Godzina Odjazdu</label>
  <input type="datetime-local" name="departure_time" id="departure_time" class="form-control @if ($errors-
>first('departure_time')) is-invalid @endif required>
  <div class="invalid-feedback">Godzina odjazdu jest wymagana.</div>
</div>
<div class="form-group mb-2">
```

```

<label for="arrival_time" class="form-label">Godzina Przyjazdu</label>

<input type="datetime-local" name="arrival_time" id="arrival_time" class="form-control @if ($errors-
>first('arrival_time')) is-invalid @endif" required>

<div class="invalid-feedback">Godzina przyjazdu jest wymagana i musi być późniejsza niż godzina
odjazdu.</div>

</div>

<div class="form-group mb-2">

<label for="ticket_price" class="form-label">Cena</label>

<input type="number" step="0.01" name="ticket_price" id="ticket_price" class="form-control @if ($errors-
>first('ticket_price')) is-invalid @endif" required min="0">

<div class="invalid-feedback">Cena biletu jest wymagana i musi być liczbą nieujemną.</div>

</div>

<div class="form-group mb-2">

<label for="stops" class="form-label">Przystanki (Przystanki muszą być oddzielone przecinkiem)</label>

<input type="text" name="stops[]" id="stops" class="form-control @if ($errors->first('stops')) is-invalid
@endif" required>

<div class="invalid-feedback">Przystanki są wymagane.</div>

</div>

```

Walidacja po stronie klienta i servera:

- Pole bus number jest wymagane.
- Pole distance jest wymagane.
- Pole departure city jest wymagane.
- Pole arrival city jest wymagane.
- Pole departure time jest wymagane.
- Pole arrival time jest wymagane.
- Pole ticket price jest wymagane.
- Pole stops.0 musi być ciągiem znaków.

Dodaj Przejazd

Numer Autobusu

Numer autobusu jest wymagany i nie może przekraczać 255 znaków.

Dystans (km)

Dystans jest wymagany i musi być liczbą nieujemną.

Miasto Wyjazdu

Miasto wyjazdu jest wymagane i nie może przekraczać 255 znaków.

Choose File No file chosen

Miasto Przyjazdu

Miasto przyjazdu jest wymagane i nie może przekraczać 255 znaków.

Choose File No file chosen

Godzina Odjazdu

Godzina odjazdu jest wymagana.

Godzina Przyjazdu

Godzina przyjazdu jest wymagana i musi być późniejsza niż godzina odjazdu.

Cena

Cena biletu jest wymagana i musi być liczbą nieujemną.

4. Aktualizacja przejazdu

Walidacja danych podczas aktualizacji przejazdu odbywa się w metodzie update kontrolera ScheduleController i wykorzystuje klasę UpdateScheduleRequest.

```
public function rules(): array
{
    return [
        'bus_number' => 'required|string|max:255',
        'distance' => 'required|numeric|min:0',
        'departure_city' => 'required|string|max:255',
        'arrival_city' => 'required|string|max:255',
        'departure_time' => 'required|date|date_format:Y-m-d\TH:i',
        'arrival_time' =>
        'required|date|after:departure_time|date_format:Y-m-d\TH:i',
        'ticket_price' => 'required|numeric|min:0',
        'departure_city_img' =>
        'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        'arrival_city_img' =>
        'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        'stops' => 'array',
        'stops.*' => 'string|max:255',
    ];
}
```

Walidacja jest taka sama jak przy dodawaniu nowego przejazdu.

```
<div class="form-group mb-2">
    <label for="bus_number" class="form-label">Numer Autobusu</label>
    <input type="text" name="bus_number" id="bus_number" class="form-control @if ($errors-
>first('bus_number')) is-invalid @endif" value="{{ $schedule->bus_number }}" required maxlength="255">
    <div class="invalid-feedback">Numer autobusu jest wymagany i nie może przekraczać 255 znaków.</div>
</div>
<div class="form-group mb-2">
    <label for="distance" class="form-label">Dystans (km)</label>
    <input type="number" step="0.01" name="distance" id="distance" class="form-control @if ($errors-
>first('distance')) is-invalid @endif" value="{{ $schedule->route->distance }}" required min="0">
    <div class="invalid-feedback">Dystans jest wymagany i musi być liczbą nieujemną.</div>
</div>
<div class="form-group mb-2">
    <label for="departure_city" class="form-label">Miasto Wyjazdu</label>
    <input type="text" name="departure_city" id="departure_city" class="form-control @if ($errors-
>first('departure_city')) is-invalid @endif" value="{{ $schedule->departureCities->first()->name }}" required
maxlength="255">
    <div class="invalid-feedback">Miasto wyjazdu jest wymagane i nie może przekraczać 255 znaków.</div>
    <input type="file" name="departure_city_img" class="form-control mt-2">
</div>
<div class="form-group mb-2">
    <label for="arrival_city" class="form-label">Miasto Przyjazdu</label>
```

```

        <input type="text" name="arrival_city" id="arrival_city" class="form-control @if ($errors-
>first('arrival_city')) is-invalid @endif" value="{{ $schedule->arrivalCities->first()->name }}" required
maxlength="255">

        <div class="invalid-feedback">Miasto przyjazdu jest wymagane i nie może przekraczać 255 znaków.</div>

        <input type="file" name="arrival_city_img" class="form-control mt-2">

    </div>

    <div class="form-group mb-2">

        <label for="departure_time" class="form-label">Godzina Odjazdu</label>

        <input type="datetime-local" name="departure_time" id="departure_time" class="form-control @if ($errors-
>first('departure_time')) is-invalid @endif" value="{{ $schedule->departure_time }}" required>

        <div class="invalid-feedback">Godzina odjazdu jest wymagana.</div>

    </div>

    <div class="form-group mb-2">

        <label for="arrival_time" class="form-label">Godzina Przyjazdu</label>

        <input type="datetime-local" name="arrival_time" id="arrival_time" class="form-control @if ($errors-
>first('arrival_time')) is-invalid @endif" value="{{ $schedule->arrival_time }}" required>

        <div class="invalid-feedback">Godzina przyjazdu jest wymagana i musi być późniejsza niż godzina
odjazdu.</div>

    </div>

    <div class="form-group mb-2">

        <label for="ticket_price" class="form-label">Cena</label>

        <input type="number" step="0.01" name="ticket_price" id="ticket_price" class="form-control @if ($errors-
>first('ticket_price')) is-invalid @endif" value="{{ $schedule->ticket_price }}" required min="0">

        <div class="invalid-feedback">Cena biletu jest wymagana i musi być liczbą nieujemną.</div>

    </div>

    <div class="form-group mb-2">

        <label for="stops" class="form-label">Przystanki (Przystanki muszą być oddzielone przecinkiem)</label>

        <input type="text" name="stops[]" id="stops" class="form-control @if ($errors->first('stops')) is-invalid
@endif" value="{{ implode(' ', $schedule->route->stops->pluck('name')->toArray()) }}" required>

        <div class="invalid-feedback">Przystanki są wymagane.</div>

    </div>

```

Walidacja po stronie klienta i servera:

- Pole bus number jest wymagane.
- Pole distance jest wymagane.
- Pole departure city jest wymagane.
- Pole arrival city jest wymagane.

Edytuj Przejazd

Numer Autobusu

Numer autobusu jest wymagany i nie może przekraczać 255 znaków.

Dystans (km)

Dystans jest wymagany i musi być liczbą nieujemną.

Miasto Wyjazdu

Miasto wyjazdu jest wymagane i nie może przekraczać 255 znaków.

Choose File

No file chosen

Miasto Przyjazdu

Miasto przyjazdu jest wymagane i nie może przekraczać 255 znaków.

Choose File

No file chosen

Godzina Odjazdu

Godzina Przyjazdu

Cena

Przystanki (Przystanki muszą być oddzielone przecinkiem)

Zaktualizuj Przejazd

Podsumowując, walidacja danych w projekcie „Rozkład autobusowy” zapewnia integralność danych i poprawność działania aplikacji, chroniąc przed wprowadzeniem nieprawidłowych informacji.

Podsumowanie:

Dokumentacja projektu „Rozkład autobusowy” przedstawia kompleksowy opis procesu tworzenia aplikacji, która umożliwia zarządzanie i przeglądanie rozkładów jazdy autobusów. Dzięki wykorzystaniu Laravel 11.x oraz silnika Blade, aplikacja jest modularna, łatwa do rozszerzenia i utrzymania. MySQL zapewnia solidne podstawy do przechowywania danych, a Bootstrap oraz JavaScript zapewniają interaktywność i responsywność interfejsu użytkownika. Walidacja danych jest kluczowym elementem projektu, zapewniającym integralność danych i bezpieczeństwo aplikacji. Implementacja różnych funkcjonalności, takich jak logowanie, rejestracja, zarządzanie zasobami oraz filtrowanie wyników, pokazuje praktyczne zastosowanie najlepszych praktyk programistycznych.