

Dokumentacja projektu		AI1
Autor	Piotr Sztainmiller, 117838	Data oddania
Kierunek, rok	Informatyka, II rok, st. stacjonarne (3,5-I)	12.07.2022
Specjalizacja	–	
Grupa	LAB 4	Ocena
Temat projektu	<i>Biuro turystyczne</i>	

Tematyka projektu:

Aplikacja "Biuro podróży" ma za zadanie ułatwić administrację biura podróży. Założeniem projektu jest przejrzyste, proste i szybkie dodawanie, edytowanie, usuwanie klientów/wycieczek, bez możliwości pomylenia się podczas dodawania informacji. Program wspiera dodawanie wielu wycieczek dla jednego użytkownika.

Wykorzystane technologie:

PHP (<https://www.php.net>) – interpretowany język skryptowy zaprojektowany do generowania stron internetowych oraz budowania aplikacji webowych. Główny język wykorzystywany w projekcie.

Laravel (<https://laravel.com/>) - framework do aplikacji internetowych napisany w języku PHP bazujący na wzorcu architektonicznym Model-View-Controller. Główny framework wykorzystywany w aplikacji do zaimplementowania wszystkich funkcji. Posiada silnik renderowania widoków html [Blade](#). Oraz ORM [Eloquent](#) ułatwiający obsługę relacyjnych baz danych. Obie te funkcjonalności zostały wykorzystane w projekcie.

HTML5 (<https://html.spec.whatwg.org/>) – język używany do tworzenia struktur stron internetowych.

PgAdmin – Aplikacja przeznaczona do zarządzania serwerem oraz bazami danych PostgreSQL.

Visual Studio Code (<https://code.visualstudio.com/>) – darmowy edytor kodu źródłowego z kolorowaniem składni dla wielu języków, stworzony przez Microsoft,

o otwartym kodzie źródłowym. Kod źródłowy Visual Studio Code jest wolnym oprogramowaniem opublikowanym na licencji MIT, podczas gdy oficjalne wydania Visual Studio Code są objęte licencją własnościową.

Pakiet XAMPP - wieloplatformowy, zintegrowany pakiet, składający się głównie z serwera Apache, bazy danych MySQL i interpreterów dla skryptów napisanych w językach PHP. Wykorzystywany do uruchomienia lokalnej bazy danych. Licencja pozwala na darmowe użycie programu.

Instrukcja uruchomienia aplikacji:

Wymagania:

- PHP wersja 8 lub wyższa
- MySql (np. z pakietu xampp) wersja 8.0 lub wyższa.

Uruchomienie aplikacji:

- Po pobraniu pliku 117838_projekt.zip umieszczamy go w nowo utworzonym folderze i wypakowujemy pliki.
- Instalujemy potrzebne rozszerzenia do języka PHP oraz HTML
- Wchodzimy przez cmd do lokalizacji projektu i instalujemy dodatek
- za pomocą:
 - composer install
 - Jeśli to konieczne podmieniamy domyślną konfigurację połączenia z bazą danych
- w pliku „.env”. Konfiguracja połączenia z bazą danych jest zdefiniowana
- Jeśli to konieczne podmieniamy domyślną konfigurację połączenia z bazą danych w pliku „.env”. Konfiguracja połączenia z bazą danych jest zdefiniowana od linii 14
- Uruchamiamy serwer bazy danych mysql i tworzymy bazę danych
- o odpowiedniej nazwie zdefiniowanej w pliku „.env”.
- Wykonujemy następujące komendy w cmd:
 - php artisan migrate
 - php artisan serve
- Aplikacja uruchomi się pod adresem: <http://127.0.0.1:8000/>

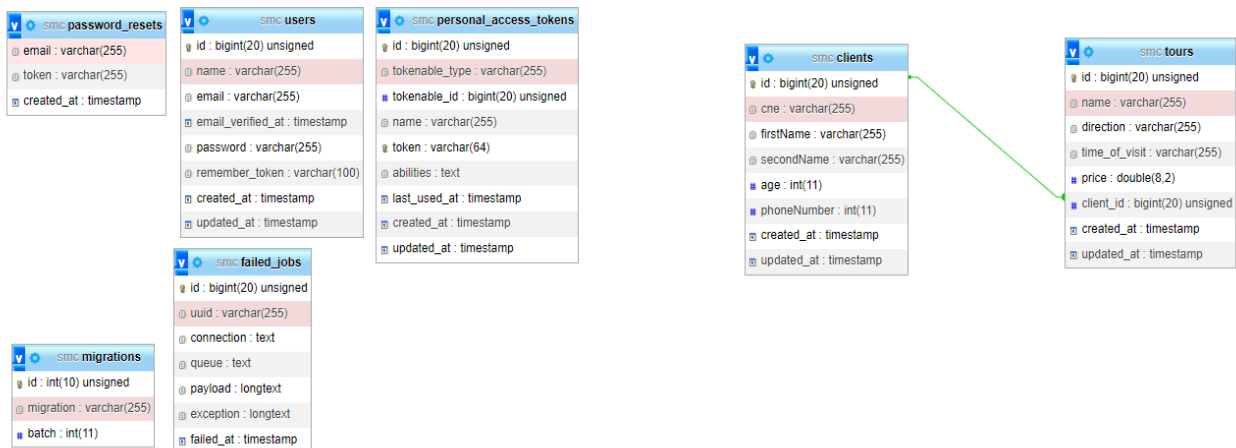
Omówienie kodu

Połączenie z bazą danych:

Konfiguracja połączenia z bazą danych znajduje się w pliku .env w 14 linii pliku. Możemy podać tam parametry takie jak: rodzaj serwera, host, port, nazwa bazy danych, nazwa użytkownika i hasło.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=smc
DB_USERNAME=root
DB_PASSWORD=
```

Tabelki:



clients - tabela przechowująca dane klienta. Kolumna id odwołuje się do client_id w tabelce tourstours - tabela przechowująca dane dotyczące wycieczki. Kolumna client_id odwołuje się do kolumny id w tabelce clients. Pozostałe tabelki są generowane automatycznie przez laravel.

1. Tabela clients:

- id,
- cne,
- firstName,
- secondName,
- age,
- phoneNumber

2. Tabela tours:

- id,
- name,
- direction,
- time_of_visit,
- price,
- client_id

Obsługa dodawania klientów wraz z walidacją:

Bardzo istotną funkcją aplikacji jest możliwość dodawania nowych klientów.

W odpowiedzi na zakładkę mamy możliwość zapoznania się z wcześniej dodanymi rekordami takimi jak imię nazwisko wiek czy numer telefonu. Dodatkowo podczas dodawania aplikacja zadba aby dane były poprawne

```

/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create()
{
    $clients = Client::all() ;
    $tours = Tour::all();
    return view('client',compact('tours','clients'),['layout'=>'create']);
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $request->validate([
        'cne'=>'required|unique:clients',
        'firstName'=>'required|string',
        'secondName'=>'required|string',
        'age'=>'required|numeric',
        'phoneNumber'=>'required|numeric|min:9'
    ]);

    $client = new Client() ;

    $client->cne = $request->input('cne') ;
    $client->firstName = $request->input('firstName') ;
    $client->secondName = $request->input('secondName') ;
    $client->age = $request->input('age') ;
    $client->phoneNumber = $request->input('phoneNumber') ;

    $client->save() ;
    return redirect('/') ;
}

```

Kolejnym przykładowym zastosowania aplikacji jest usuwanie wcześniej dodanych rekordów. Fragment kodu funkcji za to odpowiedzialnej poniżej.

```

public function destroy($id)
{
    $client = Client::find($id) ;
    $client->delete();
    return redirect('/');
}

```

Przykładowy widok:

```
<body>
@include("navbar")

<div class="row header-container justify-content-center">
  <div class="header">
    <h1>Client management system</h1>
  </div>
</div>

@if($layout == 'index')
  <div class="container-fluid mt-4">
    <div class="container-fluid mt-4">
      <div class="row justify-content-center">
        <section class="col-md-7">
          @include("clientslist")
        </section>
      </div>
    </div>
  </div>

@elseif($layout == 'create')
  <div class="container-fluid mt-4">
    <div class="row">
      <section class="col">
        @include("clientslist")
      </section>
      <section class="col">
        <div class="card mb-3">
          
          <div class="card-body">
            <h5 class="card-title">Enter the informations of the new client</h5>
            <form action = "<?php echo e(url('/store')) >?" method="post">
              @csrf
              <div class="form-group">
                <label>CNE</label>
                <input name="cne" type="text" class="form-control" placeholder="Enter cne">
              </div>
              <div class="form-group">
                <label>First Name</label>
                <input name="firstName" type="text" class="form-control" placeholder="Enter first name">
              </div>
              <div class="form-group">
                <label>Second Name</label>
                <input name="secondName" type="text" class="form-control" placeholder="Enter second name">
              </div>
              <div class="form-group">
                <label>Age</label>
                <input name="age" type="text" class="form-control" placeholder="Enter age">
              </div>
            </form>
          </div>
        </div>
      </section>
    </div>
  </div>
</div>
```

Przykładowy model:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Client extends Model
{
    use HasFactory;

    protected $table = 'clients';

    public function tours(){
        return $this->hasMany(Tour::class);
    }
}
```

Przykładowa migracja:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('tours', function (Blueprint $table) {
            $table->id();

            $table->string('name');
            $table->string('direction');
            $table->string('time_of_visit');
            $table->float('price');
            $table->unsignedBigInteger('client_id')->nullable();
            $table->foreign('client_id')->references('id')->on('clients');



            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('tours');
    }
};
```

Dodawanie klientów oraz ich przeglądanie:

Client Management System - Test Data - Add Client - Add Client

Client management system



List of clients

You can find more information about clients in the system

ID#	First name	Second Name	Age	Phone number	Year name	Operations
1123	Robert	John	12	123456789	Wyprzedaż imprezowa na Mazowsze Wyprzedaż na Mazowsze	Add Edit Delete
1234	John	Robert	18	123456789	Wyprzedaż imprezowa na Mazowsze Wyprzedaż na Mazowsze	Add Edit Delete
1345	John	Robert	17	123456789	Wyprzedaż imprezowa na Mazowsze Wyprzedaż na Mazowsze	Add Edit Delete
1456	John	Robert	16	123456789	Wyprzedaż imprezowa na Mazowsze Wyprzedaż na Mazowsze	Add Edit Delete

Enter the information of the new client

ID#

First name

Second Name

Age

Phone number

Year name

[Add](#) [Cancel](#)

Poniżej tabelki klienta oraz dodawania nowego klienta znajduje się również zapisywanie klienta do danej wycieczki (wybieramy nazwę wycieczki, identyfikator klienta, a następnie klikamy przycisk save):

List of clients

You can find more information about clients in the system

ID#	First name	Second Name	Age	Phone number	Year name	Operations
1123	Robert	John	12	123456789	Wyprzedaż imprezowa na Mazowsze Wyprzedaż na Mazowsze	Add Edit Delete
1234	John	Robert	18	123456789	Wyprzedaż imprezowa na Mazowsze Wyprzedaż na Mazowsze	Add Edit Delete
1345	John	Robert	17	123456789	Wyprzedaż imprezowa na Mazowsze Wyprzedaż na Mazowsze	Add Edit Delete
1456	John	Robert	16	123456789	Wyprzedaż imprezowa na Mazowsze Wyprzedaż na Mazowsze	Add Edit Delete

Enter the information of the new client

ID#

First name

Second Name

Age

Phone number

Year name

[Add](#) [Cancel](#)

The first name field is required.
The second name field is required.
The age field is required.
The phone number field is required.

Przy dodawaniu nowego klienta/wycieczki występuje walidacja wprowadzanych danych, jeżeli dane są niepoprawne zostanie zwrócona informacja:

Enter the informations of the new client

CNE

Enter cne

First Name

Enter first name

Second Name

Enter second name

Age

Enter age

Phone number

Enter phone number

Save Reset


- The cne has already been taken.
- The age must be a number.
- The phone number must be a number.

Możemy również osobno przeglądać tabelki tours oraz clients:

- clients:

Clients Management Table clients Table tours Edit tours Edit clients

Client managment system



List of clients

You can find here informations about clients in the system

CNE	First name	Second Name	Age	Phone number	tour name	Operations
F123	Pioter1	Jakis	12	123456789	Wycieczka krajoznawcza po Mazurach Wycieczka do Warszawy	Edit Delete
DZ5	Piotr	Działo	99	123321432	Wycieczka do Sosnowca Wycieczka do Gdyni	Edit Delete
A11	Ewa	Pierwsza	67	734505432	Wycieczka w Bieszczady	Edit Delete
J999	Jan	Dubiel	43	567182736	Wycieczka do Lublina	Edit Delete

- tours:

Clients Management Table clients Table tours Edit tours Edit clients

Client managment system



List of tours

You can find here informations about tours in the system

name	direction	time_of_visit	price	Operations
Wycieczka do Sosnowca	Sosnowiec	7Dni	1399	Edit Delete
Wycieczka krajoznawcza po Mazurach	Zakopane	7Dni	1399	Edit Delete
Wycieczka do Warszawy	Warszawa	24h	1999	Edit Delete
Wycieczka do Gdyni	Gdynia	7Dni	999	Edit Delete
Wycieczka nad morze	Sopot	3Dni	999	Edit Delete
Wycieczka w Bieszczady	Polańczyk	3Dni	645	Edit Delete