|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Uniwersytet Morski w Gdyni** | | |  | | Nr ewidencyjny: | | | |
| **Wydział Elektryczny** | | |  | | Data złożenia pracy: . | | | |
| **Katedra Telekomunikacji**  **Morskiej** | | |  | | Nr albumu:38738 | | | |
|  | | | | | | | | |
| **PRACA DYPLOMOWA**  **INŻYNIERSKA** | | | | | | | Miejsce na fotografię  4,5 x 3,5 cm | |
| Dyplomant: | **Sebastian Goyke** | | | | | | | |
| Specjalność: | **Systemy i Sieci Teleinformatyczne** | | | | | | | |
| Promotor: | **dr inż. Piotr Kaczorek** | | | | | | | Ocena: |
| Recenzent: |  | | | | | | | Ocena: |
| Egzamin dyplomowy: | | | | Data: | | | | Ocena: |
| Temat: | | | | | | | | |
| **Projekt i realizacja systemu informacyjnego dla małej firmy usługowej z obsługą powiadomień SMS, z wykorzystaniem PHP i MySQL** | | | | | | | | |
| Recenzent: | | Promotor: | | | | Dziekan: | | |
| Dyplomant: | |  | | | |  | | |
| **Gdynia 2018** | | | | | | | | |

Sebastian Goyke Gdynia, dnia 25 – 09 – 2018r.

**OŚWIADCZENIE**

Świadomy/a odpowiedzialności prawnej oświadczam, że złożona praca inżynierska pt.:

„Projekt i realizacja systemu informacyjnego dla małej firmy usługowej z obsługą powiadomień SMS, z wykorzystaniem PHP i MySQL”

została napisana przeze mnie samodzielnie.

Równocześnie oświadczam, że w pracy wykorzystano tylko cytowaną literaturę a więc praca nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. 1994, nr 24, poz. 83) oraz dóbr osobistych chronionych prawem cywilnym.

Ponadto praca nie zawiera informacji i danych uzyskanych w sposób nielegalny i nie była wcześniej przedmiotem innych procedur urzędowych związanych z uzyskaniem dyplomów lub tytułów zawodowych uczelni wyższej.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną   
na płycie CD.

Na podstawie art. 75 §2 kodeksu postępowania administracyjnego wnoszę o odebranie tego oświadczenia jako dowodu prawdziwości okoliczności w nim podatnych, przy czym jestem świadomy odpowiedzialności karnej z art. 233 §1 i §6 k.k. za złożenie fałszywego oświadczenia.

………………………………………………… *podpis*

Zawartość

[1. Wstęp 3](#_Toc530393439)

[2. Technologia wykorzystana do realizacji projektu 4](#_Toc530393440)

[2.1. PHP 4](#_Toc530393441)

[2.2. Framework Laravel 5](#_Toc530393442)

[2.3. Powiadomienia SMS 5](#_Toc530393443)

[3.Omówienie użytych narzędzi 5](#_Toc530393444)

[3.1. Composer 5](#_Toc530393445)

[3.2. Artisan 5](#_Toc530393446)

[3.3.PHPMyAdmin 5](#_Toc530393447)

[3.4 GIT 6](#_Toc530393448)

[5. Opisanie systemu informacyjnego 6](#_Toc530393449)

[5.1. Prezentacja interfejsu graficznego i jego opis 6](#_Toc530393450)

[5.2. Wzorzec projektowy 11](#_Toc530393451)

[5.3 Struktura aplikacji 12](#_Toc530393452)

[5.4. SMS API Opis 15](#_Toc530393453)

[5.5. SERWER SMS Opis 17](#_Toc530393454)

[7. Podsumowanie 18](#_Toc530393455)

# Wstęp

Dzisiejsze firmy które przechowują bardzo dużą ilość danych, potrzebują systemu do ich przechowywania jak i wyświetlania. System ten ma im pomóc w codziennej pracy. Współczesne systemy informacyjne dla firm muszą szybko i w czytelny sposób przedstawić informacje o klientach oraz informacje które dana firma potrzebuje. System powiadomień SMS jest łatwym i szybkim sposobem poinformowania klienta o jego stanie finansowym, zaległości lub ofert które dana firma wprowadza.

Celem pracy jest przedstawienie systemu informacyjnego dla małej firmy oraz obsługa powiadomień SMS. Napisanie aplikacji która będzie łączyła w szybki sposób informacje o kliencie jak i informacji potrzebnych dla firm wraz z bramką SMS. Aplikacja jest napisana w języku PHP z wykorzystaniem framework Laravel, serwis będzie korzystał z MySQL do pracy z bazą danych.

API (Application Programming Interface) jest to komunikacja miedzy aplikacjami. Jest to interfejs który komunikuje się miedzy aplikacjami. Korzysta z bibliotek oraz funkcji dostępnych przez api. Możemy wyróżnić dwa rodzaje Web serwisu REST i SOAP. REST określa zasady do tworzenia aplikacji, cechami REST API są bezstanowość, wykorzystuje zasoby, indentifikowany przez URL. REST wykorzystuje protokół http, z wykorzystaniem metod GET, POST, PUT, DELETE.

SOAP jest ustandaryzowaną metodą przeciwieństwie co do REST, ma ściśle określone zasady tworzenia systemu komunikacyjnego. SOAP tak samo korzysta z protokołu http. SOAP wykorzystuje XML do kodowania żądań i odpowiedzi.

SMS API jak i SERWER SMS udostępnia REST API. Udostępniają też dokumentacje z której w łatwy sposób można dowiedzie jak się połączyć z ich aplikacjami oraz pokazują jak napisać żądanie do aplikacji.

# 2. Technologia wykorzystana do realizacji projektu

W tym dziale przedstawione zostaną technologie którą są wykorzystywane w projekcie. Każda użyta technologia do realizacji projektu została krótko opisana w podrozdziałach.

# 2.1. PHP

PHP interpretowany skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym. PHP wykorzystywany jest do tworzenia skryptów na serwerze WWW. Często jest umieszczany wraz z kodem HTML.

Funkcjonalność PHP jest zawarta w modułach:

moduł jądra, jest to silnik PHP

moduły oficjalne, zależne od dystrybucji PHP

repozytorium PECL, są to moduły z otwartym zrodłem napisane przez programistów dla programistów

repozytorium PEAR, moduł który przedstawia grupe typów klas

Został stworzony przez Rasmusa Lerdorfa w roku 1994. W roku 1995 został udostępniony publicznie jako PHP Tools 1.0. Język ten stał sie bardzo popularny i szybko sie rozwija. Aktualna wersja PHP 7. Składnia skryptu PHP zawsze musi zaczynać sie od znaków "<?php". Jeśli w pliku znajduje sie tylko skrypt PHP nie musimy zamykać składni, natomiast jeśli znajduje sie w pliku tez kod np: HTML , skrypt musimy zamknąć znakami "?>".

Przykładowy skrypt:

<?php

$a = 7;

$b = 44;

**function** dodawanie($liczba1, $liczba2)

{

    $suma = $liczba1 + $liczba2;

**return** $wynik; //Dodajemy argumenty i zwracamy wynik

}

echo dodawanie($a, $b); // 51

?>

PHP przede wszystkim wykorzystuje obiektowe programowanie do tego wykorzystujemy klasy i obiekty.

<?php

**class** MojaKlasa {

**public** $liczba1;

**public** $liczba2;

**public** **function** dodawanie($a, $b) {

        $wynik = $a + $b;

**return** $wynik;

    }

**public** **function** odejmowanie($a, $b) {

        $wynik = $a - $b;

**return** $wynik;

    }

}

$mojobiekt = **new** MojaKlasa();

$mojobiekt->dodawanie(3, 5);

$mojobiekt->odejmowanie(10, 5);

?>

# 2.2. Framework Laravel

Laravel jeden z ja popularniejszych i najszybciej rozwijającym się framworków napisany w PHP,

bazujący na wzorcu projektowym Model-Widok-Kontroler.

Ze względu na jego prostotę oraz bogate biblioteki jak i narzędzia został użyty do stworzenia aplikacji.

Aktualna wersja 5.6 która jest wykorzystywana w aplikacji ETLINE.

Laravel jest open source-ym projektem dostępnych dla każdego. Ma bardzo dobrą i bogatą dokumentacje.

Projekt można pobrać za pomocą narzedzia Composer. Odrazu widać w strukturze plików, że są one poukladane według wzorcja MVC.

Kontrolery mieszcza sie w katalogu Controllers , widoki oraz pliki ktore sa wykorzystywane do widoków znajduja se w resources w katalogu views.

Larvel kożysta dla widoków z szablonów blade.php. Modele zaś w folderze app. Laravel udostepnia bardzo dużo razwiazanych problemów , jest osony plik do routes, validacji,

takze katalog w ktorym umieszczamy pliki z tlumaczeniami. Dzieki plika migracji mozemy tworzyc strukture bazy danych.2.3. MySQL

SQL służy do komunikacji z bazą danych która jest relacyjna. Stworzony w 1974 roku przez IBM.

Język ten jest rozwijany aż do dnia dzisiejszego. Najbardziej powszechnymi zadaniami języka SQL jest tworzenie, odczytywanie, aktualizacja i usuwanie danych z bazy danych(CRUD).

Na potrzeby realizacji aplikacji został wykorzystany serwer MySQL, oraz Phpmyadmin to graficznego przedstawienia struktury bazy danych.

# 2.3. Powiadomienia SMS

W dzisiejszych czasach wysyłanie wiadomosci sms jest codzienniością.

Dla firm powiadomienia sms jest sobosobem przypomnienia dla swoich pracownikow jak i klientów.

Jeden z najczesciejszych powodów wysylania sms-ów jest przypomnienie o zaległej płatnosci klienta, lub wyslania oferty dla kliena.

Do tego celu firmy wykorzystuja api udastepniane przez inne firmy takie jak np: SMSAPI oraz SERWER SMS.

Powyżej podane firmy dostarczaja dokumentacje techniczna ktora pokazuje jak połączyć sie z aplikacja oraz pobierania danych z tej aplikacji.

SMS API jest jedna z aplikacji z której skorzystam do powiadomień sms. Do autoryzacji polaczenia się z aplikacja będzie wykorzystywany token który jest generowany w panelu administracyjnym aplikacji na strone WWW.

Serwer SMS druga do porównania aplikacja do wysyłania wiadomości sms. W przeciwieństwie do SMS API ,SERWER SMS wykorzystuje login oraz hasło do autoryzacji wysyłania żądań i zapytań aplikacji.

# 3.Omówienie użytych narzędzi

# 3.1. Composer

Composer jest narzędziem do zarządzania zależnościami w projekcie.

Dzięki niemu można instalować i aktualizować paczki potrzebne do projektu w zautomatyzowany spoposob, całą instalacje wykonuje za Ciebie composer.

Composer jest prostym programem i łatwym w obsłudze posiada tylko kilka komend.

Plik konfiguracjy jest nazwany "composer.json" w formacie json. Znajduja sie w nim wszystkie zaleznosci ktore composer ma zainstalowac lub zaktualizowac.

W pliku ustalamy tez jaka wersje bilboteki maja byc instalowane w projekcie wiec nie musimy wybierac najnowszej wersji,

ktorej juz nie beda mogly byc uruchomione ze wzgedu na np: starsza wersje PHP , jest to bardzo przydatne do wspierania aplikacji ktore posiadaja starsze technologie.

# 

# 3.2. Artisan

Narzedzie Artisan jest instalowane razem z projektem Laravel. Artisan jest interfejsem lini polecen.

Polecenia tego narzedzia pomagaja programiscie w tworzeniu aplikacji. Programista moze dzieki niemu zautomazywac swoj projekt oraz jego inizjalizacje.

Polecenie "php artisan list" pokaze nam wszystkie dostepne komendy z ktorych udastepnia nam to narzedzie.

Od tworzenia plikow migracji,seedow, kontrolerow i modeli. Artisan udastepnia tez "Scheduling" dzieki ktoremu w pliku "Kernel.php"

mozemy wpisac polecenia ktore beda wykonywane w dokladnym czasie lub naprzyklad codziennie o godzinie ustawionej przez nas.

# 3.3. PHPMyAdmin

PHPMyAdmin jest o narzedziem w graficznym srodowisku do zarzadzania bazą danych MySQL napisana w PHP. Narzedzie to pozwala nam do tworzenia, usuwania baz danych.

Również do edycji strukury tabel, dodawnia relacji miedzy tabelami.Możemy rowniez importowac i eksportowac cała bazą danych. Dostęp do phpMyAdmina jest z poziomu przegladarki internetowej.

Twórcą jest Tobiasa Ratschillera w 1998 roku.

# 3.4 GIT

Twórcą GITa jest Linusa Torvaldsa fiński programista. Ze wzgledu na dużą liczbe twórców Linuxa oraz

liczbe folderów oraz plików które zawierał Linus Torvalds stworzył rozproszony system kontroli wersji jakim jest GIT.

Dla dużych projektów GIT jest idelanym rozwiązaniem, narzedzie to umożliwa zapisywanie

zmian zrobionych przez programiste oraz mozliwosc cofniecia sie do poprzednich zmian w programie.

Zeschronizowanie kodu dla wszystkich programistow ktorzy pracuja na danym projekcie.

Programista tez widzi swoje zmiany jakie dokonał oraz innych twórców kodu.

Gdy zdarza sie sie bład w kodzie mozna w łatwy sposob cofnac sie do kodu w ktorym dany bład nie wystepowal.

Przy inicjalizacji gita w projekcie jest tworzony plik ".git" w ktorym wszystkie wprowadzone zmiany znajduja sie w tym pliku.

Dzieki pliku ".gitignore" możemy powiedziec GIT-owi jakie pliki nie maja być sledzone i narzedzie nie bedzie obserwowało tego pliku.

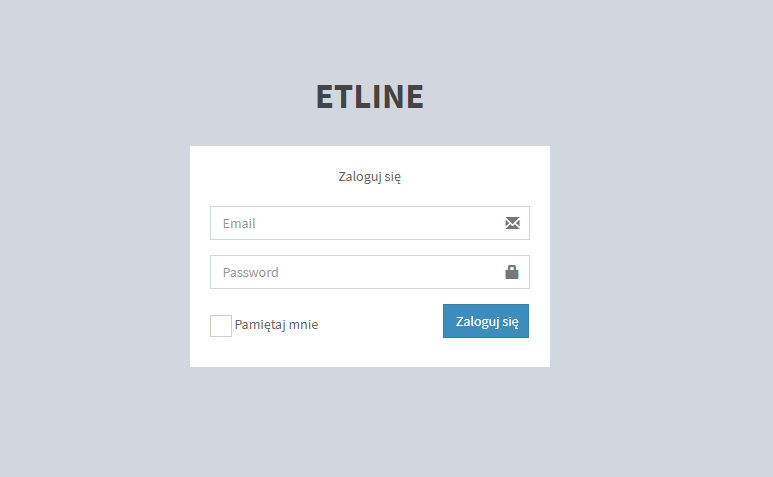
# 5. Opisanie systemu informacyjnego

# 5.1. Prezentacja interfejsu graficznego i jego opis

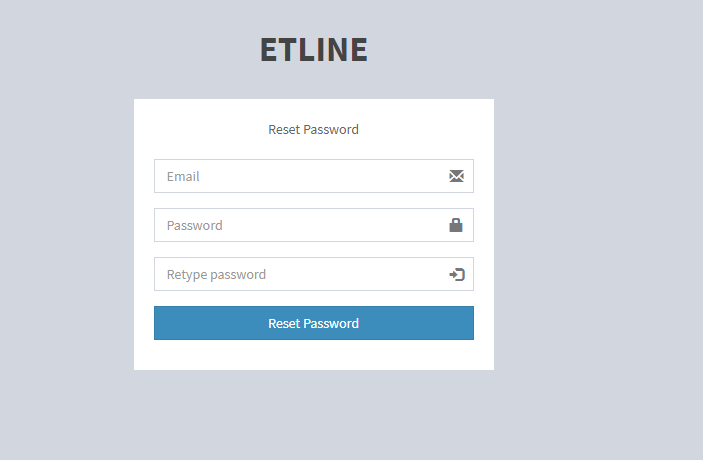
Przedstawiona aplikacja jest przykładem systemu informacyjnego dla małej firmy usługowej, zajmująca się usługami udostępniania Internetu oraz telewizji. Dodatkowo z obsługą powiadomień SMS. Interfejs graficzny jest wykorzystywany z biblioteki Admin LTE, co ułatwia szybko i prosto budowanie aplikacji.

Aby dostać się do menu głównego, trzeba wypełnić formularz logując się , dostęp maja tylko Ci użytkownicy co istnieją w bazie danych. Po wypełnieniu formularza zostajemy przekserowani na strone domową. Serwis nosi nazwę "ETLINE". Aplikacja głownie zajmuję sie

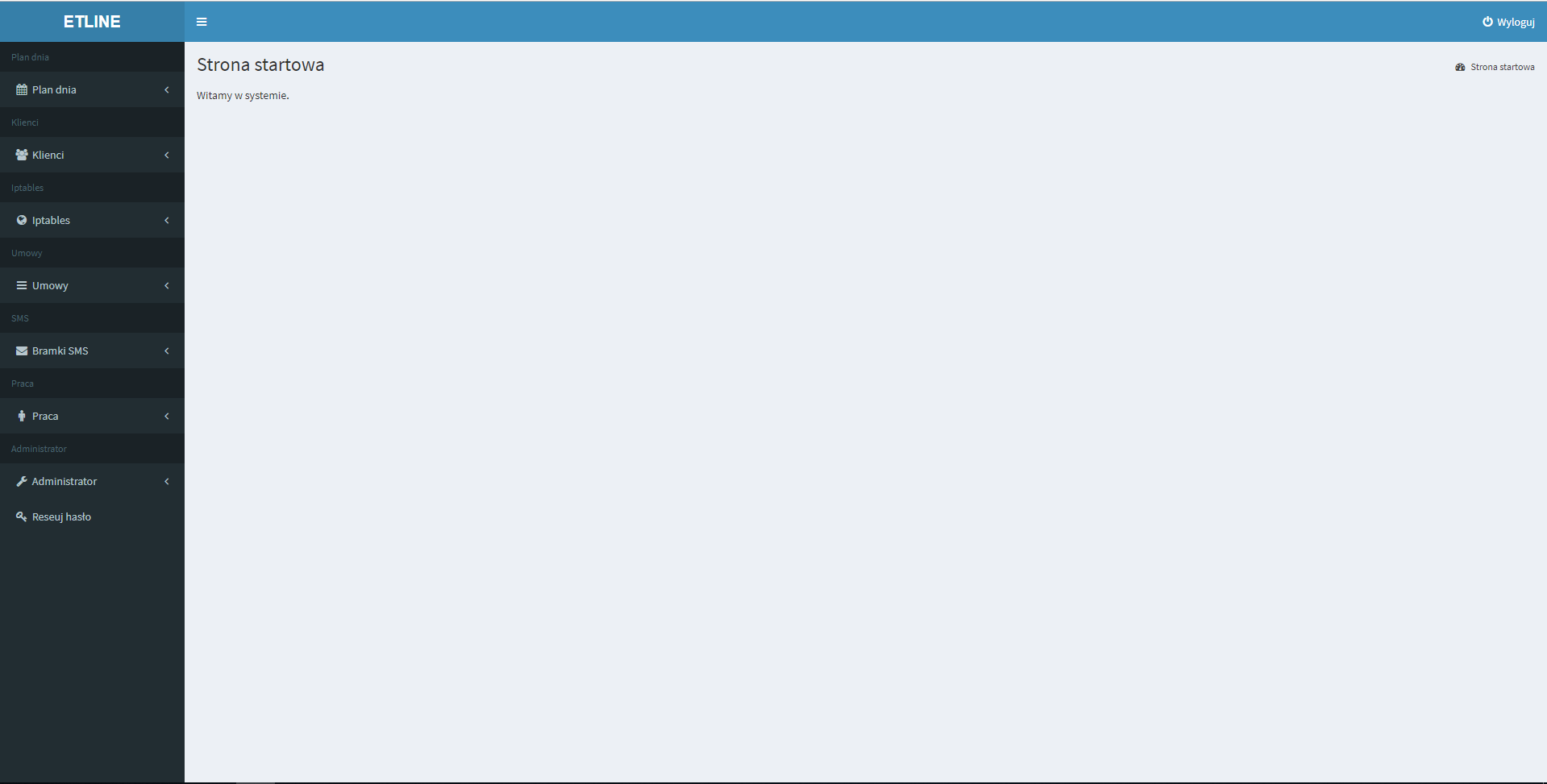
wyświetlaniem, oraz edycją danych , dzięki zakładce "Reguły”, administrator ma możliwość nadania dostępu użytkownikowi do odpowiednich zakładek.



Rys. 5.1. Formularz logowania



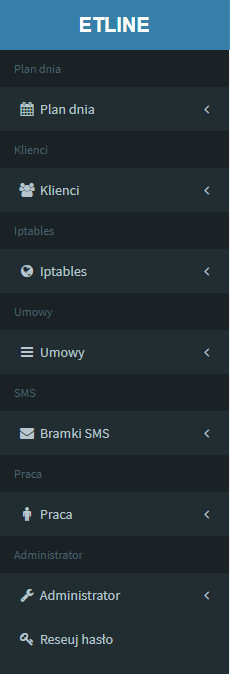
Rys. 5.2. Formularz resetowania hasła użytkownika



Rys. 5.3. Interfejs graficzny - fragment strony startowej

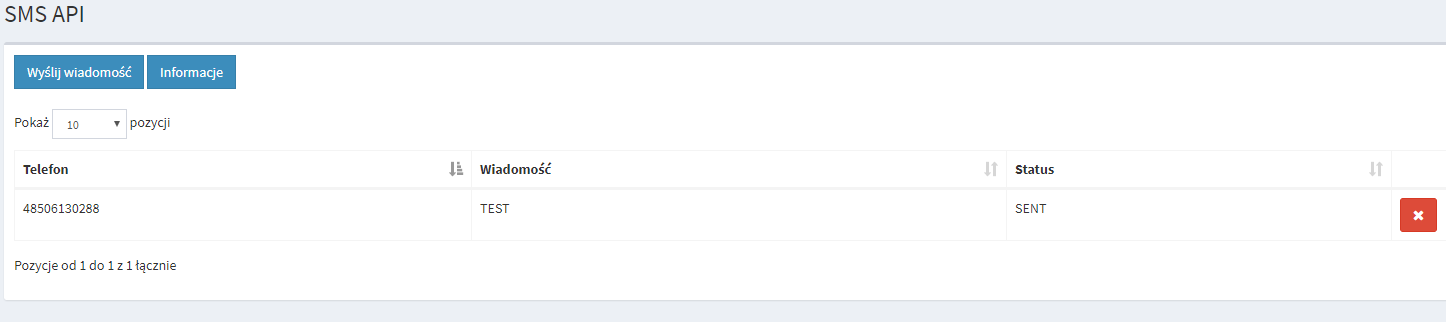
Z listy menu możemy wybrac sobie zakładke do której zostaniemy przekierwani, została ona podzielona na:

* Plan dnia
* Klienci
* Iptables,
* Umowy
* SMS
* Praca
* Administrator



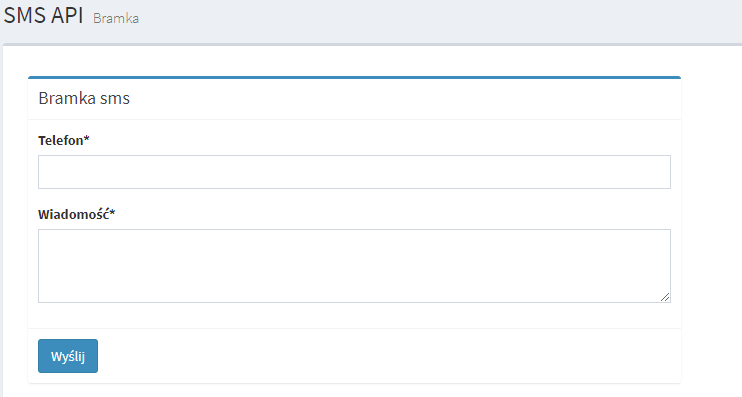
Rys. 5.4. Interfejs graficzny – Lista zakładek

W zakładce Bramki SMS, możemy wybrać sobie SMS API, po kliknieciu zostaniemy przekserowani na liste wysłanych wiadomości za pomocą aplikacji SMS API. Jak widzimy na poniższym obrazku mamy informacje na jaki numer telefonu został wysłany sms, treść wiadomości oraz status czy dana wiadomość zosała pomyślnie wysłana. Możemy także usunąć tą wiadomość z bazy danych klikając na czerwony przycisk z krzyżykiem. Na stronie widzimy też przyciski które przekieruja nas na strone Informacji o wyslanych wiadomości oraz do bramki sms z ktorej będziemy mogli wysłać wiadomość sms.



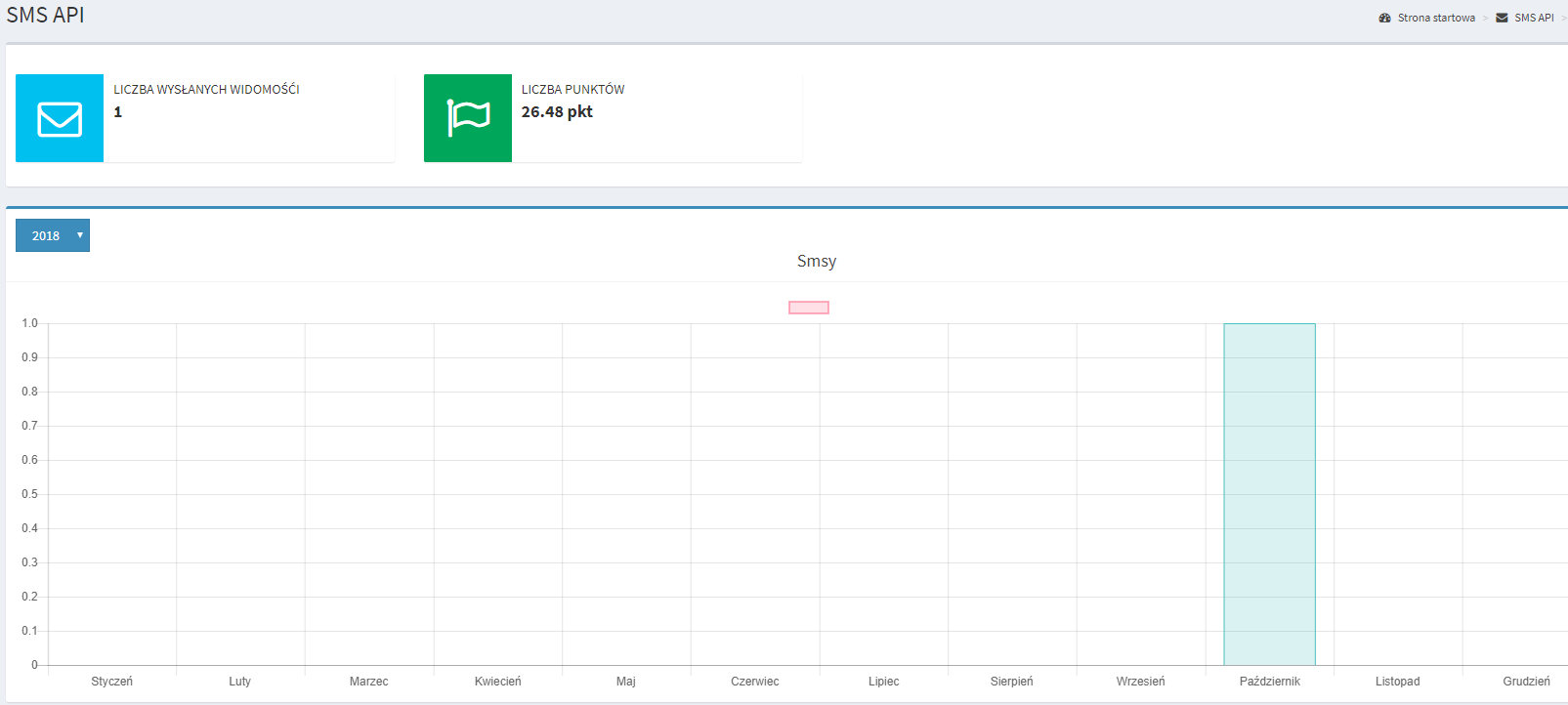
Rys. 5.5. SMS API – Lista wiadomości

Na obrazku 5.6 widzimy bramke sms dla SMS API jest to prosta i przejrzysta bramka, w ktorej wpisujemy numer telefonu i treść wiadomości i za pomocą przycisku „Wyślij” wysyłamy wiadomość do odbiorcy.



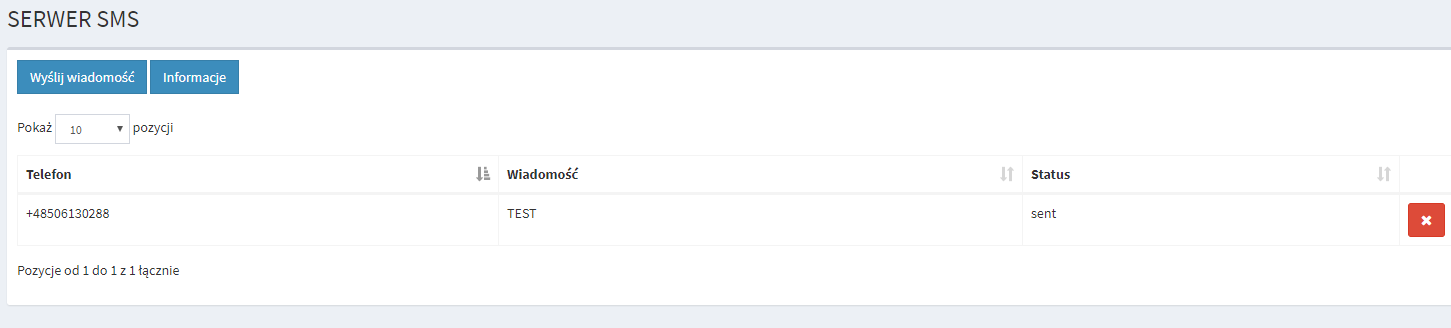
Rys. 5.6. SMS API – Bramka SMS

W zakładce „Informacje” znajdziemy , informacje o ilości wyslanych wiadomości oraz o liczbie punktów z korej możemy skorzystac do wyslania wiadomości. Na wykresie poniżej , możemy odczytac ile wiadomości zostalo wyslane w danym miesiącu i roku.



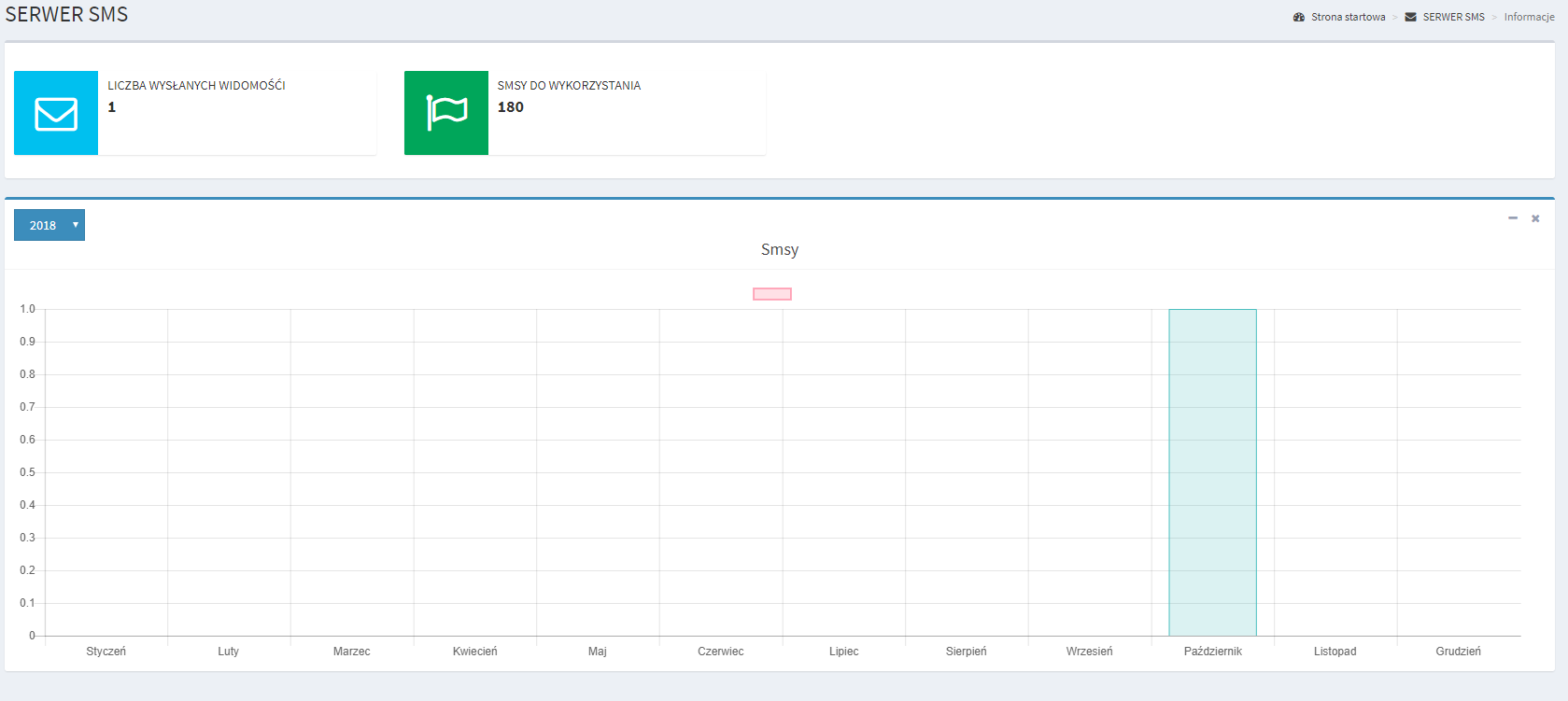
Rys. 5.7. SMS API – Informacje

Dla aplikacji SERWER SMS dostaniemy takie same informacje jak w przykładzie wczesniej. Liste wiadomości wyslanych z ich numerami oraz trescia wiadomości.



Rys. 5.8. SERWER SMS – Lista wiadomości

W przykładzie SERWER SMS w zakładce „Informacje” mam jedna roznice zamiast ilości punktów mamy ile wiadomości sms możemy jeszcze wysłać.



Rys. 5.9. SERWER SMS – Informacje



Rys. 5.10. SERWER SMS – Bramka SMS

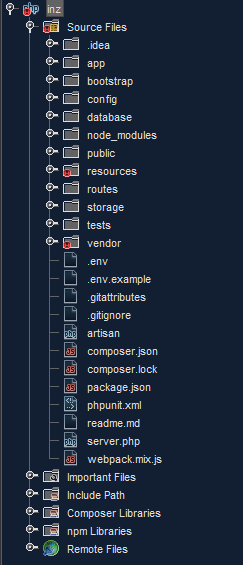
# 5.2. Wzorzec projektowy

Wzorzec projektowy można nazwać szablonem, który jest wykorzystywany w różnych sytuacjach. Jest standardem z kórego będziemy budować naszą aplikacje. Dzieki wzorcowi nasza aplikacja jest łatwiejsza do napisania, co sprawia, że kod napisany będzie przejrzysty i wydajniejszy. Model – Widok – Kontroler jest wzorcem z którego jest zbudowana aplikacja „ETLINE”.

Model –Widok – Kontroler jest aktualnie najczęściej wykorzystywanem wzorcem projektowym z którego kozystaja dzisiejsi programiści. Widok jest odpowiedzialny za wyświetlanie, prezentacje interfejsu graficznego dla użytkownika, komunikuje się z kontrolerem. Zadaniem kontrolerów jest obsługa żądań przesylanych przez serwer oraz jest tez warstwa komunikacji miedzy modelem, a widokiem. Model jest odpowiedzialny za komunikacje z bazą danych, pobierania danych oraz ich dodawania jak usuwania i edycji.

# 5.3 Struktura aplikacji

Struktura aplikacji dzięki framorkowi laravel oraz wzorzec MVC jest uporządkowana i przejrzysta dla programisty.



Przykąłdowy kontroler EmployessController kóry w konstruktorze miesci middleware auth dzieki czemu przy każdym wywołaniu żadania musi przejść przez warstwe posrednia o nazwie auth kora sprawdza czy dany użytkownik jest autoryzowany. W przykładowym kontrolerze widzimy funkcje takie jak index, create , Edit, store i update , te metody SA w każdym kontrolerze co ulatwia pisanie kodu i jego czytanie , metody te SA odpowiedzialne za wyświetlanie odpowiednich widoków oraz zapisywaniem jak edytowaniem i usuwaniem danych z bazy danych.

1. <?php
3. **namespace** App\Http\Controllers\Employees;
5. use Illuminate\Http\Request;
6. use App\Http\Controllers\Controller;
7. use Yajra\Datatables\Datatables;
8. use App\Employees;
9. use App\Http\Requests\StoreEmployees;
10. use App\Http\Requests\UpdateEmployees;
12. **class** EmployeesController extends Controller {
14. **public** function \_\_construct() {
15. $**this**->middleware('auth');
16. }
18. **public** function index() {
19. **return** view('employees/index');
20. }
22. **public** function show($id) {
23. $model = Employees::find($id);
25. **return** view('employees/show')->with('model', $model);
26. }
28. **public** function create() {
29. **return** view('employees/create');
30. }
32. **public** function edit($id) {
33. $model = Employees::find($id);
35. **return** view('employees/edit')->with('model', $model);
36. }
38. **public** function store(StoreEmployees $request) {
39. $model = $**this**->prepareDBquery($request, **new** Employees);
40. $model->save();
42. **return** redirect()->to('employees')->with('success', trans('t\_messages.content.success.add'));
43. }
45. **public** function update(UpdateEmployees $request) {
46. $model = $**this**->prepareDBquery($request, Employees::find($request->id));
47. $model->save();
49. **return** redirect()->to('employees')->with('success', trans('t\_messages.content.success.edit'));
50. }
52. **public** function **delete**($id) {
53. $model = Employees::find($id);
54. $model->**delete**();
56. **return** redirect()->to('employees')->with('success', trans('t\_messages.content.success.delete'));
57. }
59. **public** function getAll() {
60. **return** Datatables::of(Employees::select()->get())
61. ->addColumn('manage', function ($giraffe) {
62. $id = $giraffe->id;
63. $html = '<div **class**="btn-group">
64. <a href="/employees/show/' . $id . '" type="button" **class**="btn btn-default"><i **class**="fa fa-search"></i></a>
65. <a href="/employees/edit/' . $id . '" type="button" **class**="btn btn-default"><i **class**="fa fa-edit"></i></a>';
67. $html .= '<button href="/employees/delete/' . $id . '" type="button" class="btn btn-danger" onClick="modalConfirm($(this).attr(\'href\'))"><i class="fa fa-remove"></i></button>';
69. $html .= '</div>';
71. **return** $html;
72. })
73. ->rawColumns(['manage'])
74. ->make(**true**);
75. }
77. **public** function prepareDBquery(Request $request, Employees $model) {
78. $model->name = $request->name;
79. $model->position = $request->position;
80. $model->phone = $request->phone ? $request->phone : '';
82. **return** $model;
83. }
85. }

Laravel wykorzystuje pliki migracyjne do tworzenia struktury bazy danych przykładowy plik poniżej przedstawia strukure tabeli „users” w ktorej podajemy nazwy kolumn oraz i właściwości.

1. <?php
3. use Illuminate\Support\Facades\Schema;
4. use Illuminate\Database\Schema\Blueprint;
5. use Illuminate\Database\Migrations\Migration;
7. **class** CreateUsersTable extends Migration {
9. /\*\*
10. \* Run the migrations.
11. \*
12. \* @return void
13. \*/
14. **public** function up() {
15. Schema::connection('mysql')->create('users', function (Blueprint $table) {
16. $table->increments('id');
17. $table->string('name', 100);
18. $table->string('email', 50)->unique();
19. $table->string('password', 255);
20. $table->boolean('admin')->**default**(**false**);
21. $table->integer('config\_id')->unsigned();
22. $table->rememberToken();
23. $table->timestamps();
24. });
25. }
27. /\*\*
28. \* Reverse the migrations.
29. \*
30. \* @return void
31. \*/
32. **public** function down() {
33. Schema::connection('mysql')->dropIfExists('users');
34. }
36. }

Na obrazku poniżej mamy przedstawiony przykładowy model w którym ustawiamy z jaka baza danych się łączymy i podajemy tabele która będzie przedstawiona ta klasa modelu w tym przykładzie łączymy się z baza „mysql\_manager” z której wybieramy tabele „employees”.

1. <?php
3. namespace App;
5. **use** Illuminate\Database\Eloquent\Model;
7. **class** Employees **extends** Model
8. {
9. **protected** $connection = 'mysql\_manager';
11. **protected** $table = 'employees';
12. }

# 5.4. SMS API Opis

W kontrolerze SmsApiController dla obsługi bramki SMS API mamy metode "info" która pobiera z bazy danych ilość wysłanych danych w danym roku ,którry wybraliśmy w widoku.

Dodatkowo przez bilbioteke UserFactory łączymy sie z api i pobieramy liczbe punktów która jest wyświetlana w widoku.

1. **public** **function** info() {
2. $client = Client::createFromToken(env('SMSAPI\_TOKEN'));
3. $years = SmsApi::select(DB::raw('YEAR(created\_at) as year'))->distinct('year')->get();
4. $count = SmsApi::all()->count();
6. $smsapi = **new** UserFactory;
7. $smsapi->setClient($client);
9. try {
10. $points = $smsapi->actionGetPoints()->execute()->getPoints();
11. } catch (SmsapiException $exception) {
12. $points = '';
13. }
15. **return** view('sms/smsapi/info')
16. ->with('points', $points)
17. ->with('count', $count)
18. ->with('years', $years);
19. }

Po przez zapytania ajax wywołujemy funkcje w kontrolerze „infoData” która zwracam nam tablice z wartoścami ilośći wyslanych smsów w danym miesiącu.

1. $('#years').change(**function** () {
2. $.ajax({
3. type: "post",
4. url: '/smsapi/info/data',
5. data: {
6. year: $('#years').val()
7. },
8. success: **function** (data) {
10. dataChart = data[0];
11. months = data[1];
12. }
13. }).done(**function** (data) {
14. rawChart();
15. });
16. });
18. $('#years').change();
19. **public** **function** infoData(Request $request) {
20. $months = [
21. trans(\_\_('t\_common.months.January')),
22. trans(\_\_('t\_common.months.February')),
23. trans(\_\_('t\_common.months.March')),
24. trans(\_\_('t\_common.months.April')),
25. trans(\_\_('t\_common.months.May')),
26. trans(\_\_('t\_common.months.June')),
27. trans(\_\_('t\_common.months.July')),
28. trans(\_\_('t\_common.months.August')),
29. trans(\_\_('t\_common.months.September')),
30. trans(\_\_('t\_common.months.October')),
31. trans(\_\_('t\_common.months.November')),
32. trans(\_\_('t\_common.months.December'))];
33. $arr = [];

36. **for** ($i = 1; $i <= 12; $i++) {
37. $data = SmsApi::select('created\_at')->whereMonth('created\_at', '=', $i)->whereYear('created\_at', '=', $request->year)->count();
39. array\_push($arr, $data);
40. }
42. **return** [$arr, $months];
43. }

# 5.5. SERWER SMS Opis

Drugą aplikacja do wysyłania powiadomień sms została wykorzystana aplikacja „Serwer SMS”.

Tak samo jak w przypadku Sms Api mamy ten sam widok. Róznica miedzy jednym a drugim jest widoczna w kodzie.

1. **public** **function** info() {
2. $years = SerwerSms::select(DB::raw('YEAR(created\_at) as year'))->distinct('year')->get();
3. $count = SerwerSms::all()->count();
5. try {
7. $serwersms = **new** SMS;
8. $r = $serwersms->account->limits();
10. $limit = $r->items[1]->value;
11. } catch (Exception $e) {
12. $limit = '';
13. }
15. **return** view('sms/serwersms/info')
16. ->with('years', $years)
17. ->with('count', $count)
18. ->with('limit', $limit);
19. }

Do wysyłania wiadomości sms wykorzystujemy klase SMS dostępną przez api Serwer SMS.

Wywołujemy metode „sendSms” kórej podaje my parametry do tablicy , takie jak numer telefonu, treść wiadomości oraz nazwe nadawcy. Po odpowiedzi z serwera api dostajemy informacje które są zapisywane do bazy danych a te z kolei są wyświetlane w tabeli w widoku.

1. **public** **function** store(StoreSerwersms $request) {
2. try {
4. $serwersms = **new** SMS;
6. $result = $serwersms->messages->sendSms(
7. **array**(
8. '+48' . $request->phone
9. ), $request->text, 'ETLINE', **array**(
10. 'details' => true,
11. )
12. );
14. **foreach** ($result->items **as** $sms) {
16. $model = **new** SerwerSms;
17. $model->sms\_id = $sms->id;
18. $model->phone = $sms->phone;
19. $model->text = $sms->text;
20. $model->save();
21. }
22. } catch (Exception $e) {
23. **return** redirect()->to('serwersms');
24. }
26. **return** redirect()->to('serwersms')->with('success', trans('t\_messages.content.success.add'));
27. }

# 7. Podsumowanie

Celem zaprezentowanej pracy było zaprojektowanie i realizacja systemu informacyjnego dla małej firmy usługowej wraz z powiadomieniami SMS. Część praktyczną autor stworzył aplikacje w języku PHP z wykorzystaniem Framework Laravel wraz z obsługą bazy danych, dla systemu powiadomień SMS autor wykorzystał aplikacje SMS API oraz SERWER SMS dla porównania obsługi i korzystania z ich aplikacji.

System jest okrojoną wersją całośći, przedstawia tylko najważniejsze czesci projeku, ale wystarczająco pokazuje bardzo dobry sposób wykorzystania jezyka PHP do relazacji podobnych projektów serwisu informacyjnego.

Wykorzystanie narzedzi które zostały opisany w pracy, pokazuja jak można łatwo ułatwic prace w tworzeniu oprogramowania.

**WYKAZ LITERATURY**

1. [Zandstra Matt](https://livro.pl/autor/506493/Zandstra+Matt.html).: PHP. Obiekty, wzorce i narzędzia. Wydanie Wydawnictwo Helion 2017
2. [Terry Matula](https://helion.pl/autorzy/terry-matula).: Laravel. Tworzenie aplikacji. Receptury. Wydanie Wydawnictwo Helion
3. [Lorna Jane Mitchell](https://helion.pl/autorzy/lorna-jane-mitchell) API nowoczesnej strony WWW. Usługi sieciowe w PHP Wydanie Wydawnictwo Helion