POLITECHNIKA
RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Aplikacje internetowe

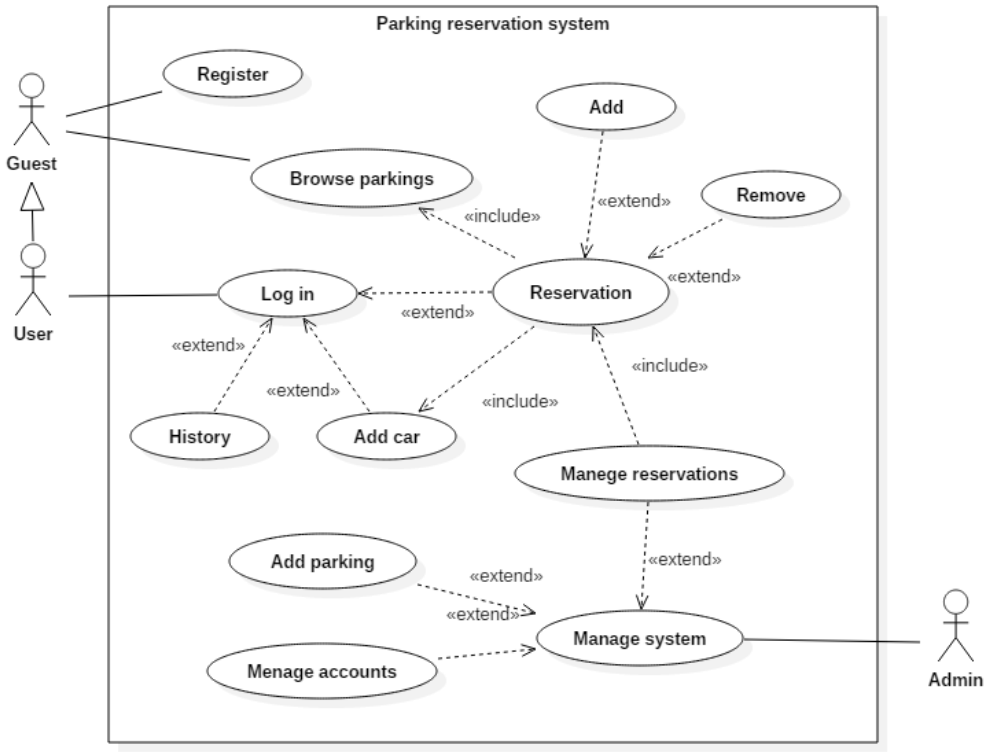
System rezerwacji miejsc parkingowych

Michał Tokarczyk
Rafał Wajszczuk
F2C – DU L7
2015-12-08

1. Wstęp

Nieustanny rozwój miast powiązany jest ze wzrostem natężenia ruchu ulicznego. O ile czynione są prace nad poprawą upłynnienia ruchu ulicznego poprzez rozbudowę dróg, to sektor parkingowy jest zaniedbywany. Zaparkowanie auta w centrum miasta bądź przy większych centrach handlowych w godzinach szczytu sprawia poważne problemy. Aby to usprawnić i dać możliwość użytkownikowi ruchu drogowego pewność, że po dojechaniu do celu będzie mógł bezpiecznie zaparkować swój samochód powstał system rezerwacji miejsc parkingowych.

Celem projektu było opracowanie systemu zarządzającego miejscami parkingowymi na specjalnie przystosowanych w tym celu parkingach. Cały system został podzielony na dwie części. Pierwsza z nich umożliwia zarezerwowanie wolnych miejsc postojowych poprzez serwis on-linie. Potencjalny użytkownik logując się do systemu posiada możliwość wyboru konkretnego parkingu, który aktualnie go interesuje. Następnie zostanie wyświetlona lista przedstawiająca aktualny stan parkingu, z pośród dostępnych wolnych miejsc może dokonać rezerwacji na swoje dane. Drugą częścią systemu jest instalacja elektroniczna zainstalowana na specjalnie przystosowanych w tym celu parkingach. Dostarcza ona niezbędnych informacji dla całego systemu informując kierowców a także zarządcę nadzorującego pracę o aktualnym statusie poszczególnych miejsc parkingowych. Do każdego stanowiska przypisana jest tablica informacyjna zawierająca numer miejsca, jego aktualny status oraz jeśli jest zarezerwowana to przez jakiego użytkownika.



rys. 1. Funkcjonalności - diagram przypadków użycia

2. Oprogramowanie

WebServ 2.1

Czołowy, Polski, darmowy pakiet wolnego oprogramowania (WAMP) do obsługi witryn internetowych dla systemu Windows. Zawiera najnowsze wersje programów:

rys. 2. WebServ 2.1

- Apache 2.2.22 (VC9)
- PHP 5.4.10 (VC9) + PDFlib-8.0.5
- MySQL 5.5.21
- Perl 5.14.2
- CesarFTP 0.99g
- Webalizer 2.1
- No-IP Dynamic Update Client 3.0.4
- DynDNS Update Client 4.1.10

MySQL Workbench 6.2.3

MySQL Workbench to narzędzie do zarządzania i modelowania baz danych MySQL. Za jego pomocą można edytować konfigurację serwera i jego komponentów, a także zaprojektować i stworzyć schematy (wizualne reprezentacje tabel, widoków itp.) nowych baz danych, wykonać dokumentację istniejących oraz zapewnić wsparcie przy procesach migracji do MySQL.

rys. 3. MySQL Workbench 6.2.3

Główne cechy narzędzia to oparcie o silnik graficzny OpenGL, wsparcie dla procesów reverse-engineering'u i synchronizacji baz danych, możliwość generowania skryptów SQL, przeglądowy tryb pracy w którym cały model bazy jest prezentowany w jednym przekrojowym widoku, wsparcie dla projektowania baz na poziomach koncepcyjnym, logicznym i fizycznym, rozszerzalna architektura, eksport modelu jako skryptu SQL typu CREATE, import i eksport modeli DBDesigner4, wizualna reprezentacja tabel, widoków, procedur wbudowanych oraz funkcji oraz pełne wsparcie dla możliwości MySQL 5.

NetBeans IDE 8.0.2

rys. 4. NetBeans IDE 8.0.2

NetBeans IDE jest środowiskiem programistycznym (Integrated Development Environment - IDE) - narzędziem służącym do tworzenia, kompilowania, uruchamiania i testowania programów. Całe IDE jest napisane w Javie, jednak umożliwia ono również tworzenie programów w innych językach. Aplikacja pozwala znacznie przyspieszyć pisanie kodów źródłowych programów, stron WWW oraz usług sieciowych, dzięki zastosowaniu licznych funkcji. NetBeans IDE oferuje między innymi liczne generatory, podpowiadanie oraz kolorowanie składni. Ponadto dostępna jest duża ilość modułów (plugins) rozszerzających jego możliwości.

FileZilla 3.11


rys. 5. FileZilla 3.11

FileZilla to prosty w obsłudze, polskojęzyczny i darmowy klient FTP o otwartym kodzie źródłowym. Obsługuje protokoły FTP, SFTP i FTPS (FTP po SSL/TLS). Główne zalety programu FileZilla to wznawianie przerwanych połączeń (jeśli serwer je obsługuje), wykrywanie bezczynności, wsparcie dla zapory sieciowej, wsparcie Proxy SOCKS4/5 i HTTP 1.1, obsługa połączeń szyfrowanych, definiowane reguły ograniczania szybkości wysyłania i odbierania, możliwość kopiowania plików do i z serwera FTP metodą „przeciągnij i upuść", definiowanie własnych komend, menedżer witryn, kolejka pobierania i wysyłania, konfigurowalny interfejs i wiele innych.

Framework'i

Bootstrap 3.3.5


rys. 6. Bootstrap 3.3.5

Bootstrap to obecnie jeden z najpopularniejszych szkieletów do tworzenia atrakcyjnych interfejsów stron WWW. Jest to framework CSS, wydany na licencji MIT. Zawiera zestaw przydatnych narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Bazuje głównie na gotowych rozwiązaniach HTML oraz CSS (kompilowanych z plików Less) i może być stosowany m.in. do stylizacji takich elementów jak teksty, formularze, przyciski, wykresy, nawigacje i innych komponentów wyświetlanych na stronie. Framework korzysta także

z języka JavaScript. Projekt zapoczątkowany przez programistów Twittera powstał w 2010 roku i zyskał licznych zwolenników ze względu na bogactwo komponentów, świetną dokumentację i zgodność z różnymi przeglądarkami internetowymi. Nie bez znaczenia jest również fakt, że Bootstrap wspiera budowę responsywnych stron WWW.

CodeIgniter 3.0.0

CodeIgniter jest to framework napisany w języku PHP przez Ricka Ellisa, implementujący wzorzec Model-View-Controller. Dostarcza on zestawu narzędzi dla osób, budujących aplikacje internetowe za pomocą PHP. Umożliwia rozwój projektów znacznie szybciej niż pisanie kodu od podstaw, poprzez bogaty zestaw bibliotek dla najczęściej potrzebnych zadań, jak również posiada prosty interfejs i logiczną strukturę dostępu do tych bibliotek. CodeIgniter pozwala twórczo skupić się na projekcie, minimalizując ilość kodu potrzebnego dla danego zadania.

rys. 7. CodeIgniter 3.0.0

AngularJS 1.3.15

AngularJS jest otwartą biblioteką języka JavaScript, wspierana i firmowana przez Google, wspomagająca tworzenie i rozwój aplikacji internetowych na pojedynczej stronie. Zadaniem biblioteki jest wdrożenie wzorca Model-View-Controller (MVC) do aplikacji internetowych, aby ułatwić ich rozwój i testowanie.

rys. 8. AngularJS 1.3.15

3. Program

Aplikacja składa się z wielu komponentów współpracujących ze sobą. Zasadniczym elementem jest framework CodeIgniter, który po wstępnej konfiguracji staje się w zasadzie mostem pośredniczącym pomiędzy AngularJS,a bazą danych MySQL. Funkcjonalność CodeIgniter'a można zaobserwować przy stronie z formularzami logowania/rejestracji i resetowania hasła. Natomiast pozostałą część aplikacji obsługuje AngularJS. Zastosowanie Bootstrap'a jako warstwy front-end'u pozwoliło na szybkie przejście do pisania zasadniczej części aplikacji nie skupiając się za bardzo na arkuszach styli CSS.

CodeIgniter

Stworzone zostały dwa kontrolery, „main" oraz „user", które zarządzają modelami „parking_model" oraz „user_model". Pierwszy jest odpowiedzialny za przetwarzanie danych dotyczących parkingów, natomiast drugi zarządza stroną logowania użytkownika.

Kontroler „main" (**Listing *1***.) po załadowaniu modelu, w domyślnej metodzie „index" udostępnia użytkownikowi trzy widoki podzielone logicznie na nagłówek, treść oraz stopkę. Treść zawiera tablicę parkingów pobraną z modelu. Pozostałe metody w zasadzie są niezbędne Angular'owi.

Listing 1. Fragment konstrolera "main"

```
class Main extends CI_Controller
{
      public function __construct()
      {
            parent::__construct();
           $this->load->model('Parking_Model');
      }

     public function index()
     {
           $this->load->view('header');
           $this->load->view('main', array("parking" =>
                           $this->Parking_Model->get_parkings()));
           $this->load->view('footer');

     }

   // ...
```

AngularJS wywołując asynchroniczne żądanie dotyczące usunięcia rezerwacji, tak naprawdę powoduje wywołanie metody działającej na modelu „parking_model". Natomiast każda informacja zwrotna z kontrolera przekazywana jest w formacie JSON, gdyż jest to naturalny sposób opisu obiektów w Javascript (**Listing *2***.).

Listing 2. Niektóre metody kontrolera "main"

```
// ...

function removeReservation($placeID)
{
      $this->Parking_Model->removeReservation($placeID);
}

function getUsers()
{
      echo json_encode($this->Parking_Model->getUsers());

}

// ...
```

Mechanizmy CodeIgniter'a umożliwiły w prosty sposób zarządzanie sesją oraz mechanizmami operującymi na formularzach. Jednak mimo wszystko drugi kontroler („user") jest również dość rozbudowany i dla zachowania spójności niniejszego opracowania, jego kod zostanie pominięty, natomiast cały projekt z kompletnym kodem źródłowym znajduje się w załączniku (płycie CD).

Widoki dostępne dla tego kontrolera to: „add", „login", „reset" oraz współdzielone z „main" widoki: „header" i „footer".

AngularJS

Aplikacja Angular'owa również składa się z wielu segmentów tj. routing, zapytania asynchroniczne, bind'owanie danych, dyrektywy, itd..

Na samym początku tworzony jest nowy moduł o nazwie „app" i wstrzykiwana jest zależność od modułu „ngRoute" (***Listing 3.***). Następnie konfigurowany jest routing adresów i tworzony jedyny kontroler „parking". Definiowane są w nim zależności od zakresu, modułu adresacji, zapytań asynchronicznych, routing'u i opóźnień czasowych. Następnie arbitralnie przydzielana jest możliwość rezerwacji miejsca parkingowego oraz czyszczony jest obiekt „reservation" dostępny w zakresie. Metoda „getParkings" pobiera zarówno dostępne parkingi, jak i użytkowników, korzystając przy tym z „bazy, szkieletu", jaki udostępnił CodeIgniter w postaci kontrolera „main" i metod „getParkings" oraz „getUsers" .

Listing 3. Fragment aplikacji bazującej na framework'u AngularJS

```
var app = angular.module('app', ['ngRoute']);

app.config( function ( $routeProvider ) {
     $routeProvider
     .when('/',                  {templateUrl:  'main/parkings'})
     .when('/parking/:parkingID',{templateUrl:  'main/places'})
     .when('/user_config',       {templateUrl:  'main/user_config'
                                 })
     .otherwise(                 {redirectTo:   '/'});
});

app.controller('parking', function($scope, $location, $http,
                                   $routeParams, $interval) {
     $scope.canReserv = true;
     $scope.reservation = {};

     $scope.getParkings = function(){
          $http.get('main/getParkings').success(function(data){
               $scope.parkings = data;
          });
          $http.get('main/getUsers').success(function(data){
               $scope.users = data;
          });

     };
```

Co sekundę następuje również odświeżenie danych wysyłając odpowiednie zapytanie do serwera dotyczące konkretnego aktualnie wyświetlonego parkingu (Listing 4.).

Listing 4. Uaktualnianie danych

```
// ...
     $interval(function(){
          $scope.checkReserv($routeParams.parkingID);
     }, 1000);

     $scope.checkReserv = function(parking){
          $http.get('main/getPlaces/' + (parking - 1))
               .success(function(data){
                    $scope.places = data;
                    $scope.canReserv = true;
                    $scope.places.forEach(function(place){
                         if ($scope.canReserv)
                              if (place.user_id ==
                                  $scope.user.id &&
                                  place.id_parking == parking)
                                   $scope.canReserv = false;
                    }); }); }; // ...
```

Widoki

Nadrzędny widok ładowany przy każdorazowym wyświetleniu strony użytkownikowi to „header” (Listing 5.). Cała strona napisana jest w konwencji HTML5, wobec czego dokument został oznaczony odpowiednim tagiem („doctype”). Po ustaleniu strony kodowania znaków oraz tytułu strony następuje szereg powiązań z arkuszami styli oraz skryptami Javascript. Dyrektywa ng-app powoduje powiązanie elementu DOM „body” z utworzonym wcześniej modułem „app”.

Listing 5. Zawartość pliku "header.php"

```
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Parking</title>
            <link rel="stylesheet"
href="<?=base_url('res/bootstrap/css/bootstrap.min.css');?>" />
            <link rel="stylesheet"
href="<?=base_url('res/bootstrap/css/font-awesome.min.css')?>" />
            <link rel="stylesheet"
href="<?=base_url('res/css/application.min.css')?>" />
            <link rel="stylesheet"
href="<?=base_url('res/css/style.css')?>" />
            <script
src="<?=base_url('res/js/jquery.min.js')?>"></script>
            <script
src="<?=base_url('res/bootstrap/js/bootstrap.min.js')?>"></script>

            <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angula
r.min.js"></script>
            <script src="<?=base_url('res/js/app.js')?>"></script>

            <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.15/angular
-route.js"></script>
      </head> <body class="background-dark" ng-app="app">
```

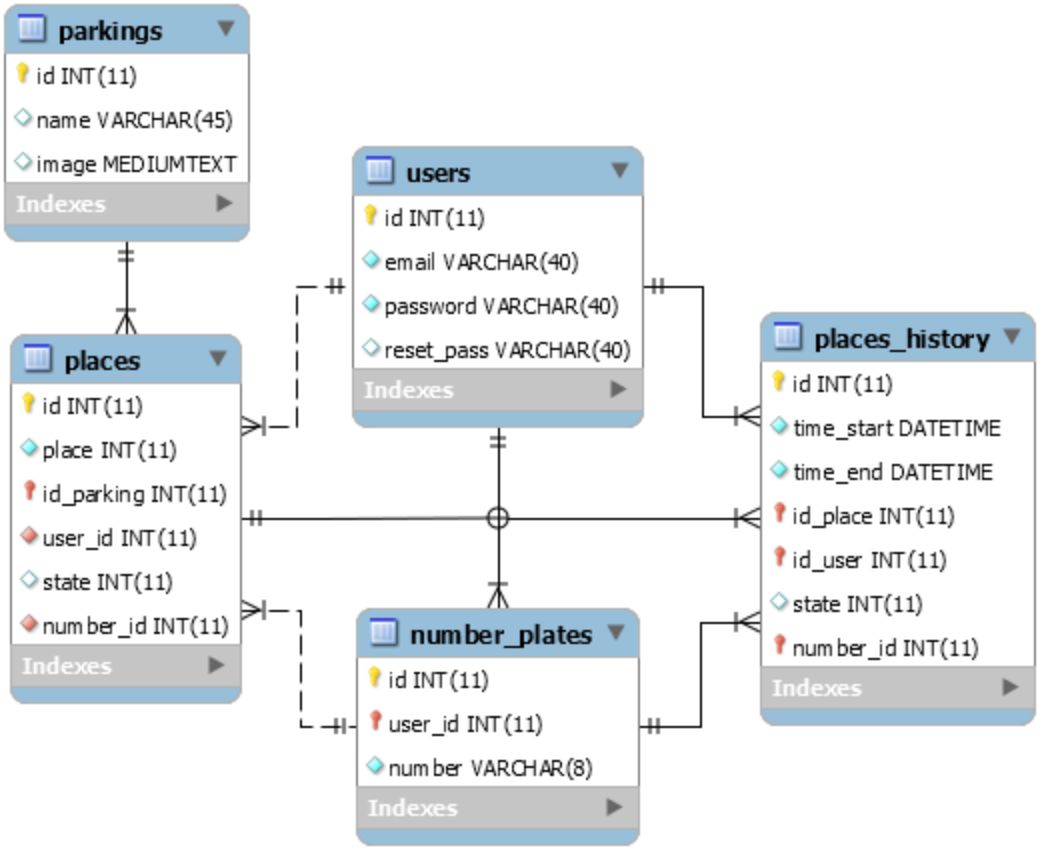
Kolejny widok, dynamicznie dołączany z routing’u „parkings” (Listing 6.) jest niezwykle zwięzły, a na jego zawartość składa się jedynie powielany znacznik „img”, którego atrybuty „wypełniane” są przez Angular’a.

Listing 6. Zawartość pliku "parkings.php"

```
<img ng-src="{{parking.image}}" alt="{{parking.name}}" title="{{parking.name}}" ng-
repeat="parking in parkings" class="img-rounded" height="250px" ng-
click="setParkingPage(parking.id)" />
```

4. Baza danych

Projekt bazy danych składa się z pięciu tabel, które połączone za pomocą odpowiednich relacji umożliwiają na elastyczne zarządzanie danymi. Wszystkie tabele posiadają klucz podstawowy o nazwie „id” i niektóre klucze obce tworzące zależności pomiędzy nimi.



rys. 9. Diagram ERD obrazujący zależności pomiędzy tabelami

Users

Aplikacja bazuje m.in. na tabeli „users”, która zawiera informacje o zarejestrowanych użytkownikach, przechowując ich adres e-mail (będący jednocześnie loginem) oraz hasło zaszyfrowane za pomocą metody SHA-1. Dodatkowe opcjonalne pole „reset_pass” jest wykorzystywane, wtedy kiedy użytkownik zażąda zresetowania hasła. Nowo wygenerowany klucz jest zapisywany w tym polu i w przypadku jego aktywacji (poprzez kliknięcie w link aktywacyjny wysłany na adres e-mail użytkownika), przepisywany jest on do „password”, jednocześnie czyszcząc „reset_pass”.

Parkings

Kolejną istotną tabelą są parkingi. Występują tutaj pola, które odpowiadają za nazwę parkingu („name”) oraz adres źródłowy do obrazka („image”). Ścieżka do pliku graficznego może zarówno prowadzić do zewnętrznej lokalizacji, jak i do pliku znajdującego się na serwerze (ścieżka względna).

Places

Każdy parking ma przydzielony zestaw miejsc i to właśnie one reprezentowane są w tabeli „places”. Klucz obcy „id_parking” wskazuje przynależność danego miejsca do konkretnego parkingu w relacji wiele-do-jednego. Opcjonalne pole „user_id” oraz „numer_id” wiązane jest odpowiednio z tabelami „users” i „numer_plates” w momencie, kiedy użytkownik o danym numerze rejestracyjnym rezerwuje bieżące miejsce. „state” jest polem twój-wartościowym przyjmującym liczby z przedziału od 0 do 2, gdzie 0 określa miejsce wolne, 1 – zajęte, a 2 – zarezerwowane.

Number_plates

Wszyscy użytkownicy muszą posiadać przyporządkowany im co najmniej jeden numer rejestracyjny, gdyż za jego pomocą będzie możliwe wykonanie rezerwacji. Numer ten zapisywany jest w bazie w postaci 8-mio znakowego ciągu alfa-numerycznego.

Places_history

Ostatnia tabela nie jest bezpośrednio wykorzystywana w projekcie (realizowany jest jedynie zapis danych), jednak ma istotne znaczenie w przypadku rozbudowy systemu, gdyż jest fundamentem pod realizację terminowych rezerwacji oraz tworzenie statystyk i przeprowadzania analiz danych. Pola „time_start” i „time_end” określają czas początku i końca istnienia miejsca parkingowego w stanie „state”. Natomiast pozostałe pola tworzą relacje pomiędzy pozostałymi tabelami.

5. Działanie aplikacji

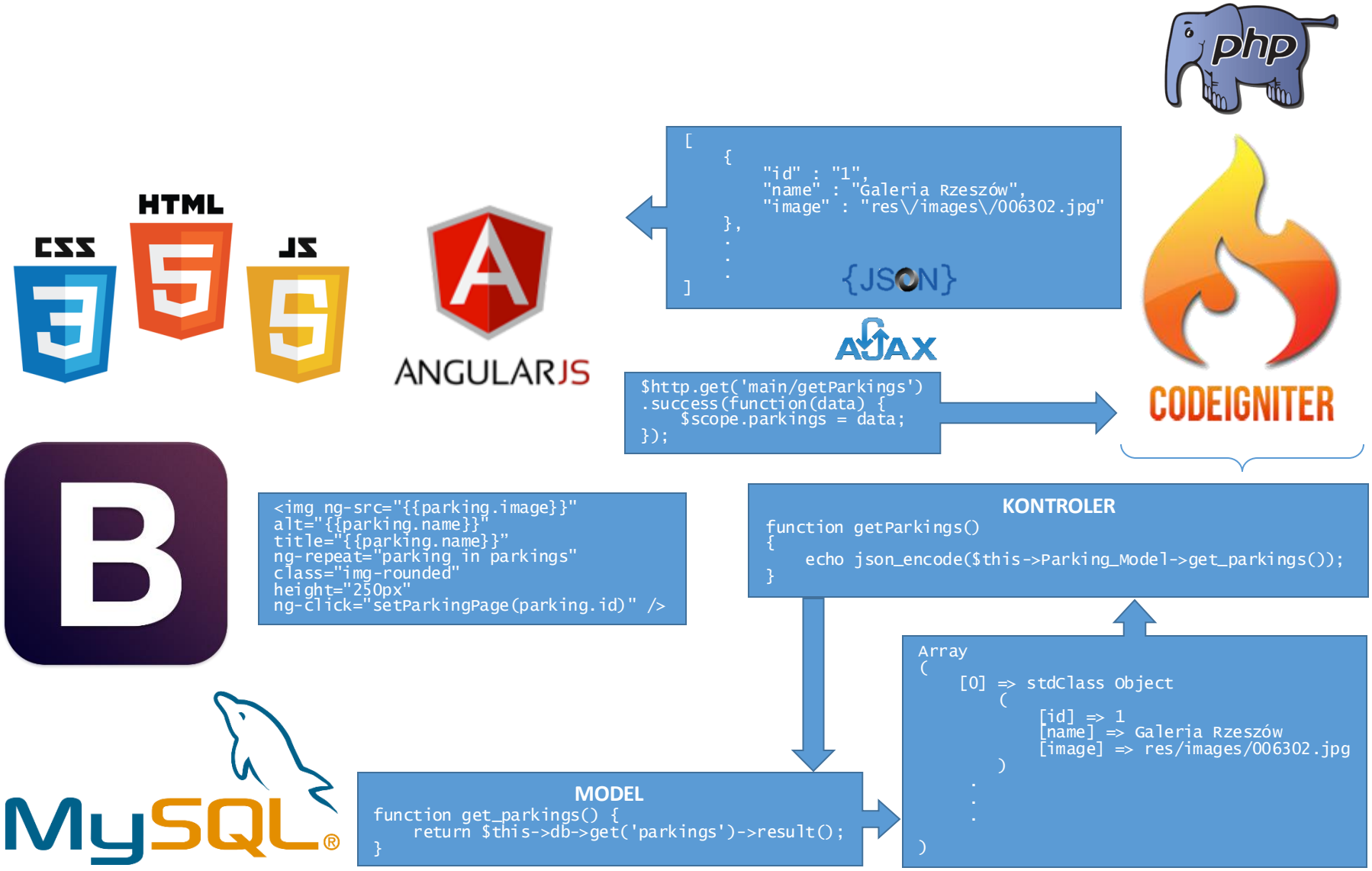
System działa w oparciu o trzy framework'i, które współpracując ze sobą w odpowiedni sposób tworzą finalną aplikację.

W opisywanej aplikacji, rola rozbudowanego funkcjonalnie framework'a CodeIgniter została ograniczona jedynie do odbierania asynchronicznych żądań od klienta (AngularJS – AJAX), pobierania lub aktualizowania informacji umieszczonych w bazie danych, odpowiedniego ich przetwarzania i zwrócenia odpowiedzi w postaci wygenerowanego widoku, bądź też kolekcji danych reprezentowanych przez format JSON. Dodatkowo CodeIgniter odpowiada również za utrzymanie sesji.

Framework AngularJS działający po stronie klienta pozwolił na stworzenie aplikacji SPA (ang. Single Page Application), która poprzez zastosowany mechanizm routingu dla użytkownika jest niezwykle wygodna w użyciu. Każda akcja wykonywana przez użytkownika, generuje odpowiednie asynchroniczne żądanie wysyłane do serwera. Informacją zwrotną jest zawsze obiekt w formacie JSON, który jest automatycznie parsowany do fizycznego obiektu JavaScirpt, w ten sposób dane w nim umieszczone mogą zostać swobodnie użyte w aplikacji.

Za minimalistyczną i niezwykle przejrzystą szatę graficzną odpowiada Bootstrap, który pozwolił w bardzo szybki sposób stworzyć warstwę prezentacyjną aplikacji. Dzięki możliwościom dynamicznego przydziału klas elementom strony, został stworzony dynamiczny interfejs użytkownika dostosowany do aktualnego stanu aplikacji.

W przykładzie przedstawionym na rys. 10. jest ukazany model przepływu danych mający za zadanie pobranie wszystkich dostępnych parkingów, w skład którego wchodzą: identyfikator, nazwa oraz ścieżka do poglądowego obrazu. W pierwszej kolejności , tuż po załadowaniu głównego kontrolera aplikacji , wysyłane jest asynchroniczne żądanie pobrania odpowiednich danych przez AngularJS do serwera na którym wywoływana jest metoda CodeIgniter'a „getParkings" kontrolera „Main". Metoda ta jest niezwykle prosta i jedynie deleguje to żądanie do modelu „Parking_Model", wywołując tym samym metodę „get_parkings", która w rezultacie swojego działania zwraca tablicę obiektów (odpowiedników zawartości tabeli „parkings" bazy danych). Wygenerowana tablica jest następnie przekształcana do formatu JSON, który jest bezpośrednią odpowiedzią dla framework'a JavaScript'owego. Pozyskane informacje w metodzie obietnicy „success" przypisywane są do obiektu w globalnym zakresie o nazwie „parkings". Ostatnim etapem jest tzw. „data-binding" tj. wiązanie danych modelu z widokiem. Za ten proces odpowiadają dyrektywy Angular'a (atrybuty z przedrostkiem „ng-"). Dyrektywa „ng-repeat" tworzy szereg znaczników w których została użyta , w tym wypadku obrazów. Każdy z nich połączony jest z konkretnym parkingiem („parking"), gdzie ich całym zbiorem jest obiekt „parkings". Tytuł oraz tekst alternatywny (dla przeglądarek tekstowych lub w przypadku nieodnalezienia obrazu do wyświetlenia) jest nazwą parkingu. Ścieżka do pliku znajduje się w polu „image", natomiast pole „id" używane jest jako parametr przekazywany funkcji „setParkingPage" odpowiedzialnej za przeniesienie użytkownika na odpowiednią stronę w ramach aplikacji SPA (ang. Single Page Application).

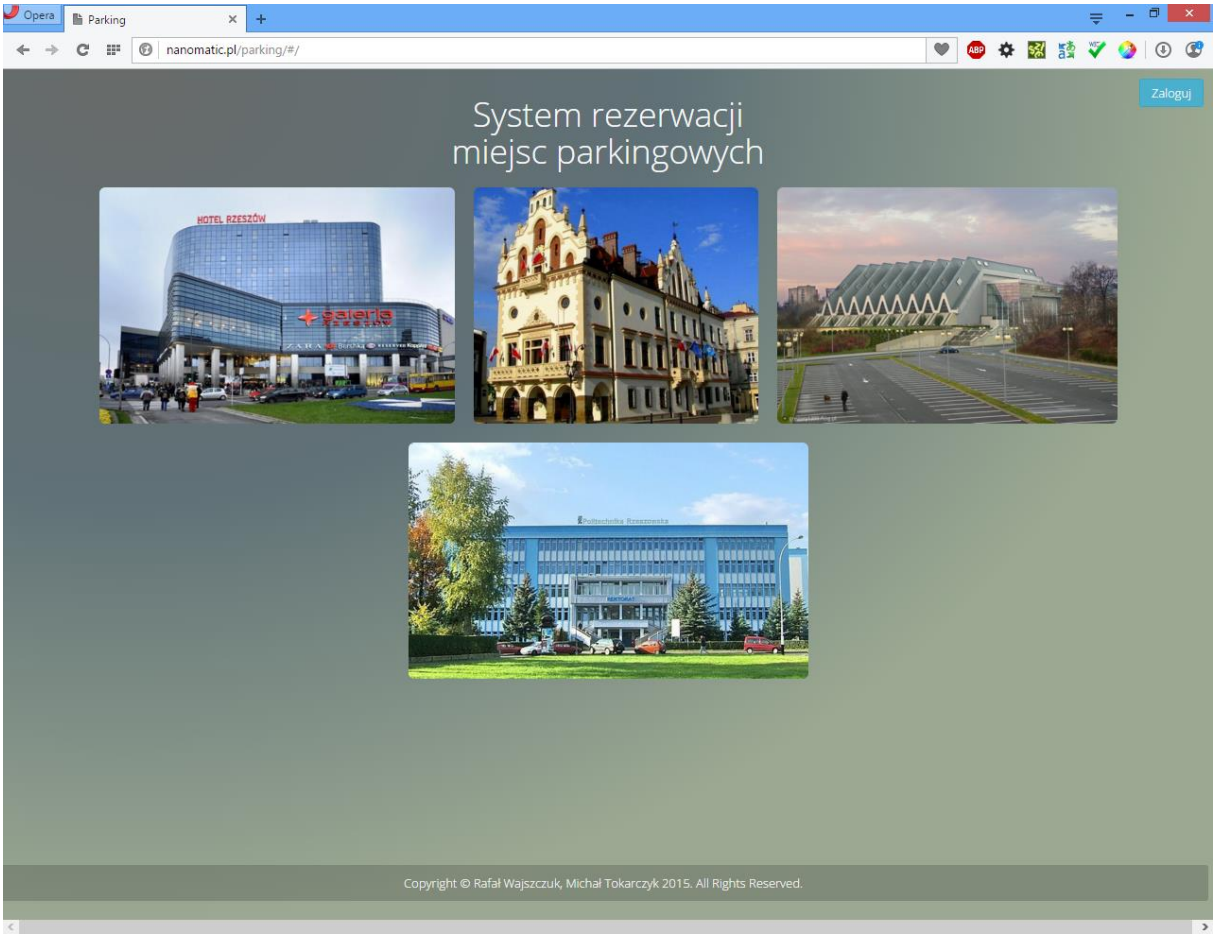


Rys. 10. Spis użytych technologii wraz z przykładem przetwarzania żądania asynchronicznego

Testowanie

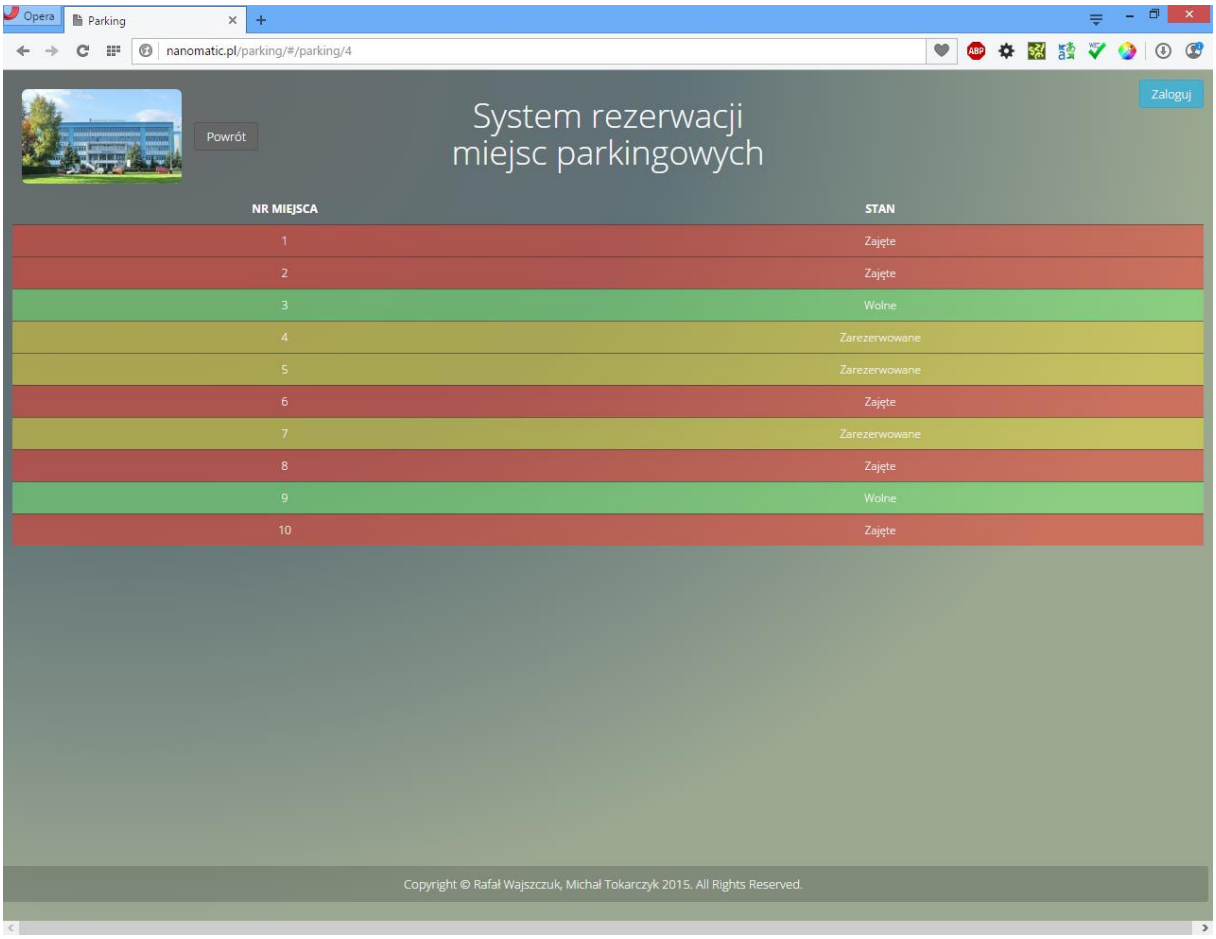
Gość

Po wejściu na stronę internetową projektu (www.nanomatic.pl/parking) wyświetlone zostają wszystkie dostępne parkingi (rysunek 11), które mogą być obserwowane i rezerwowane miejsca im przydzielone.



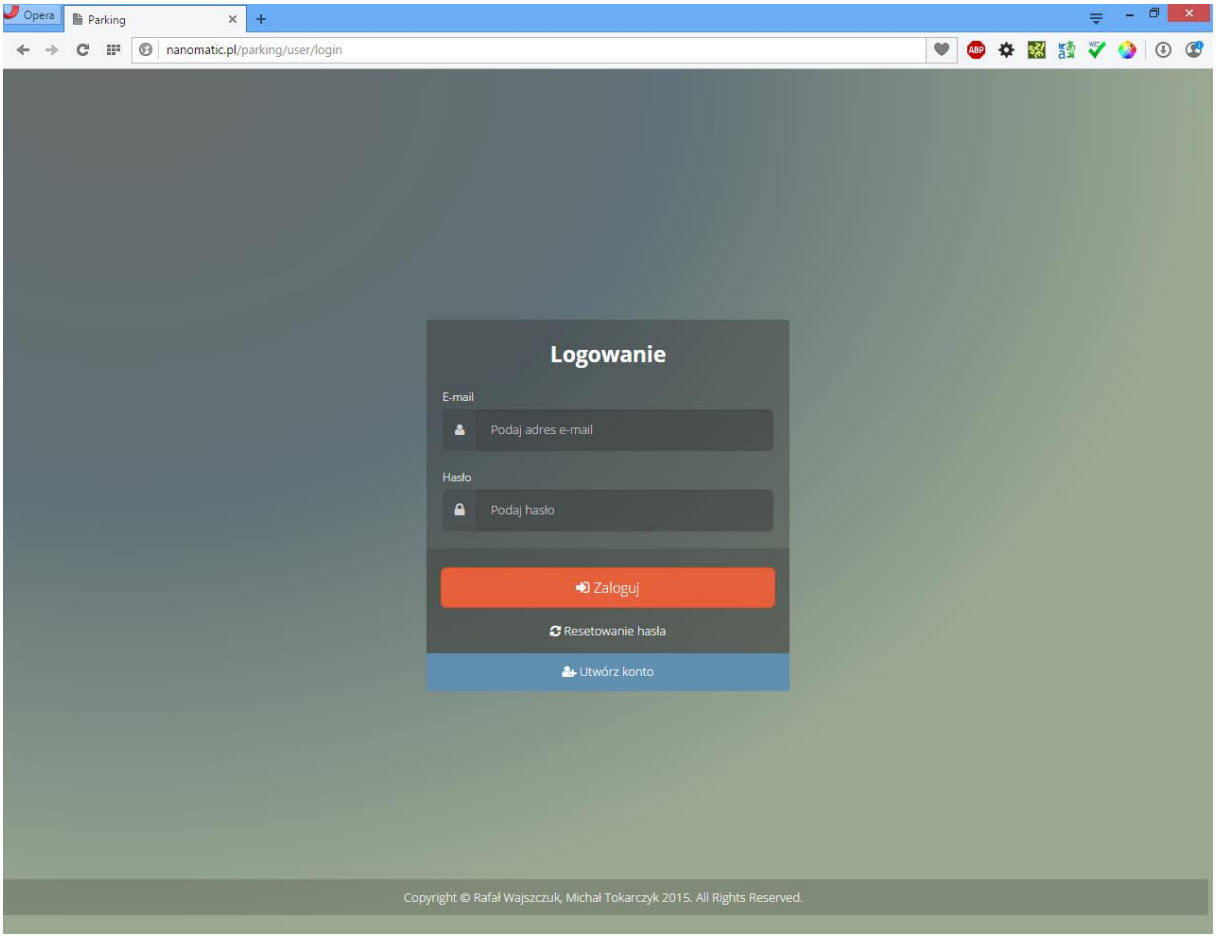
rys. 11. Ekran startowy

Wybranie dowolnego parkingu powoduje wyświetlenie listy miejsc parkingowych wraz z informacją o aktualnym ich stanie (rysunek 12). Użytkownik niezalogowany może jedynie obserwować te zmiany i nie ma fizycznej możliwości ich modyfikacji (rezerwacji).



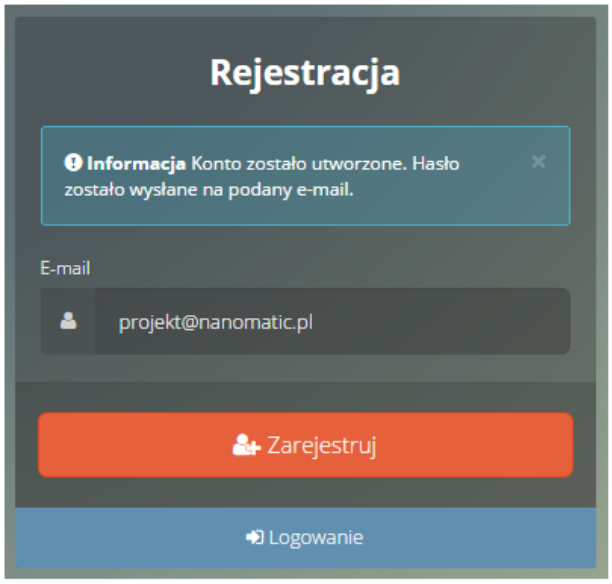
rys. 12. Widok aktualnego stanu jednego z parkingów

Naciskając przycisk zaloguj, użytkownik przenoszony jest do strony logowania (rysunek 13), gdzie po prawidłowym uzupełnieniu danych dostępowych, z powrotem następuje przekierowanie na stronę główną. Procedury rejestracji (rysunek 14a)) oraz resetowania hasła zostały uproszczone do niezbędnego minimum, a więc wymagane jest jedynie podanie swojego adresu e-mail, na który przyjdzie wiadomość z wygenerowanym 8-mio znakowym hasłem (rysunek 14b)).

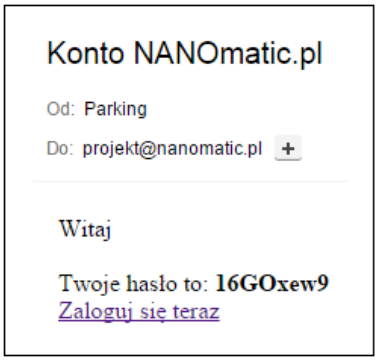


rys. 13. Strona logowania

a)



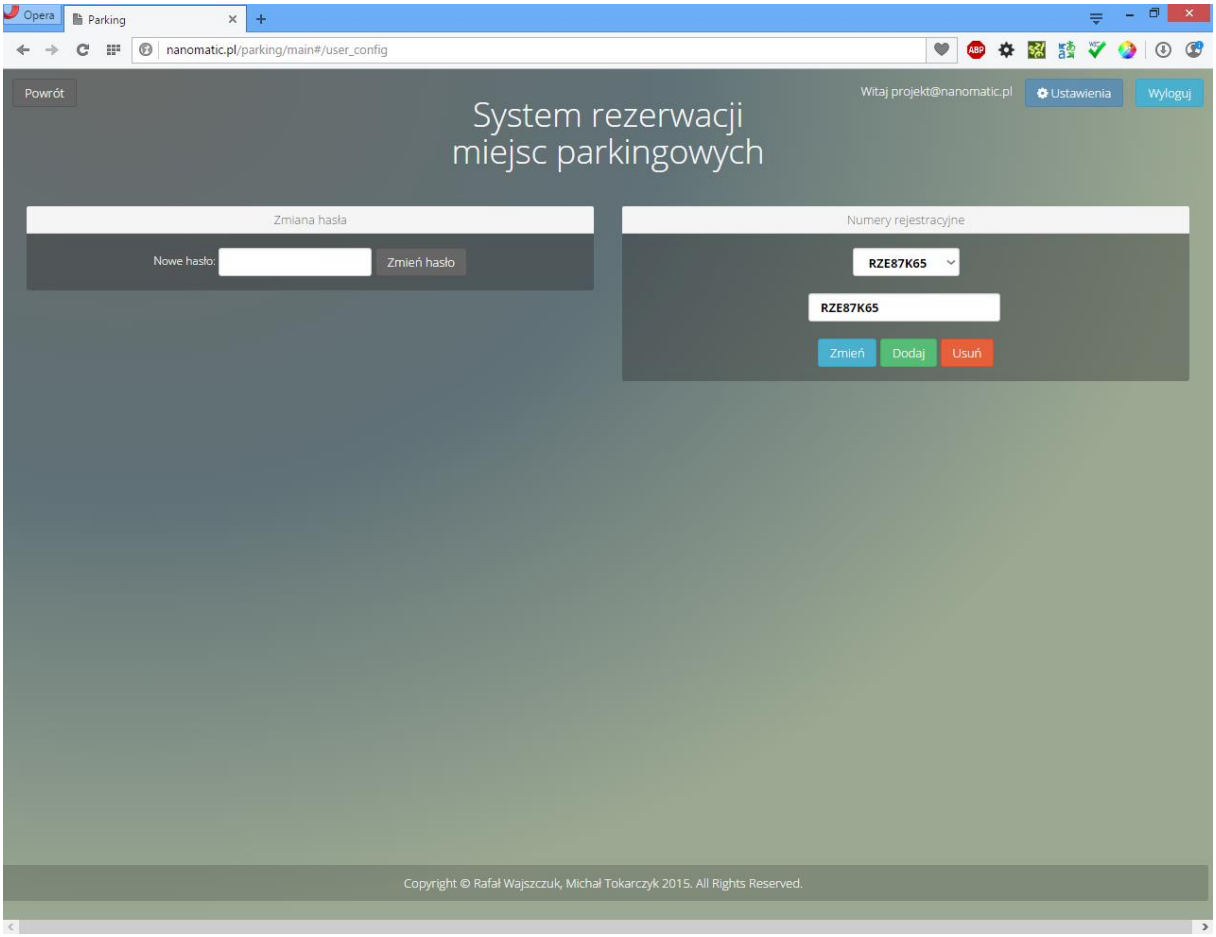
b)



rys. 14. a) Rejestracja nowego użytkownika. b) Otrzymana wiadmość e-mail

Użytkownik

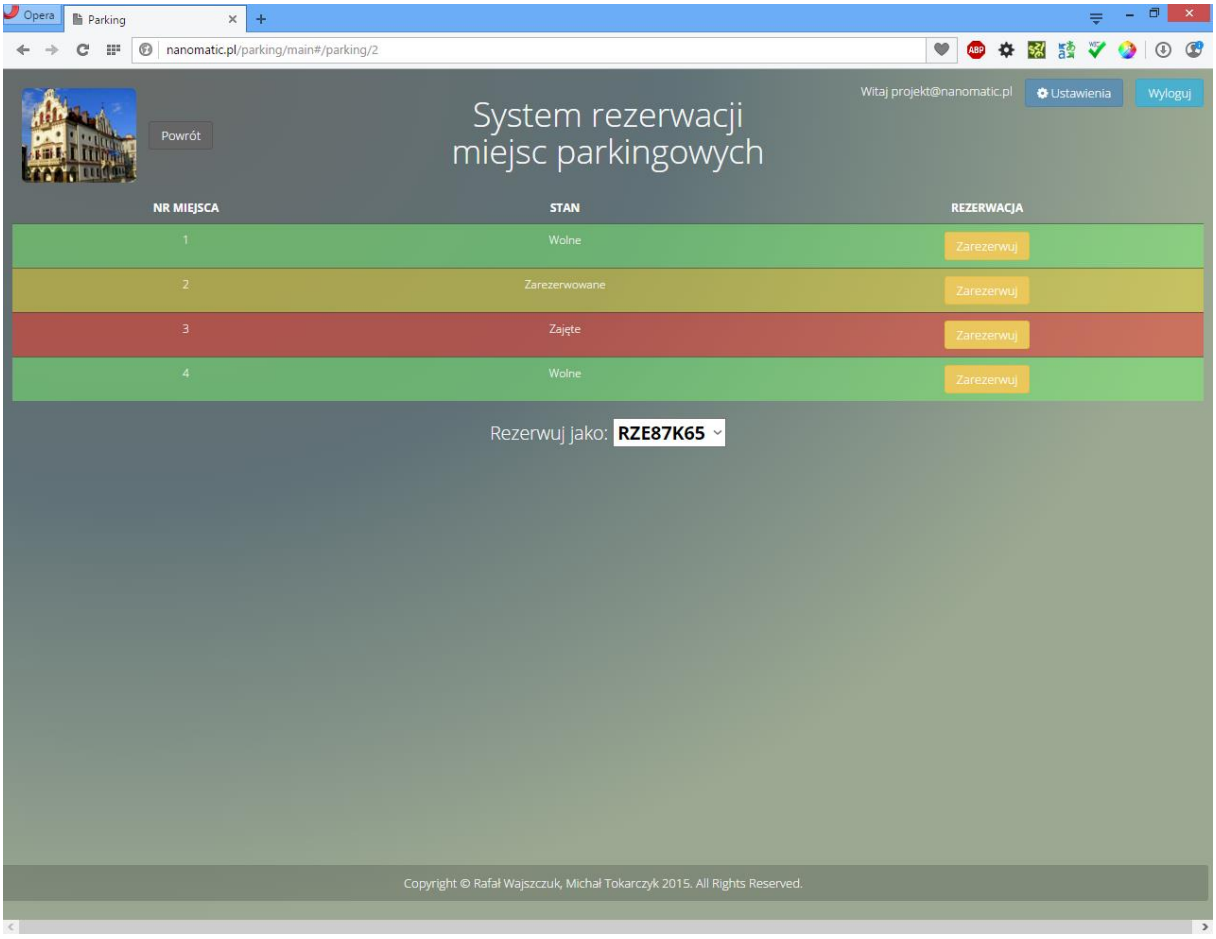
Po zalogowaniu należy przejść do ustawień (przycisk „Ustawienia” w prawym górnym rogu strony), gdzie możliwa jest zmiana hasła oraz dodanie numerów rejestracyjnych swoich samochodów (rysunek 15), na których będzie możliwe przeprowadzenie procedury rezerwacji miejsca parkingowego.



rys. 15. Ustawienia użytkownika

Po uzupełnieniu wszystkich niezbędnych danych można przystąpić do rezerwacji. W tym celu należy przejść na stronę interesującego parkingu, a następnie wybrać z listy (znajdującej się poniżej tabeli z miejscami) konkretny numer rejestracyjny na który ma zostać dokonana rezerwacja. Pozostaje jedynie wybrać miejsce i kliknąć przycisk „Zarezerwuj” (rysunek 16).

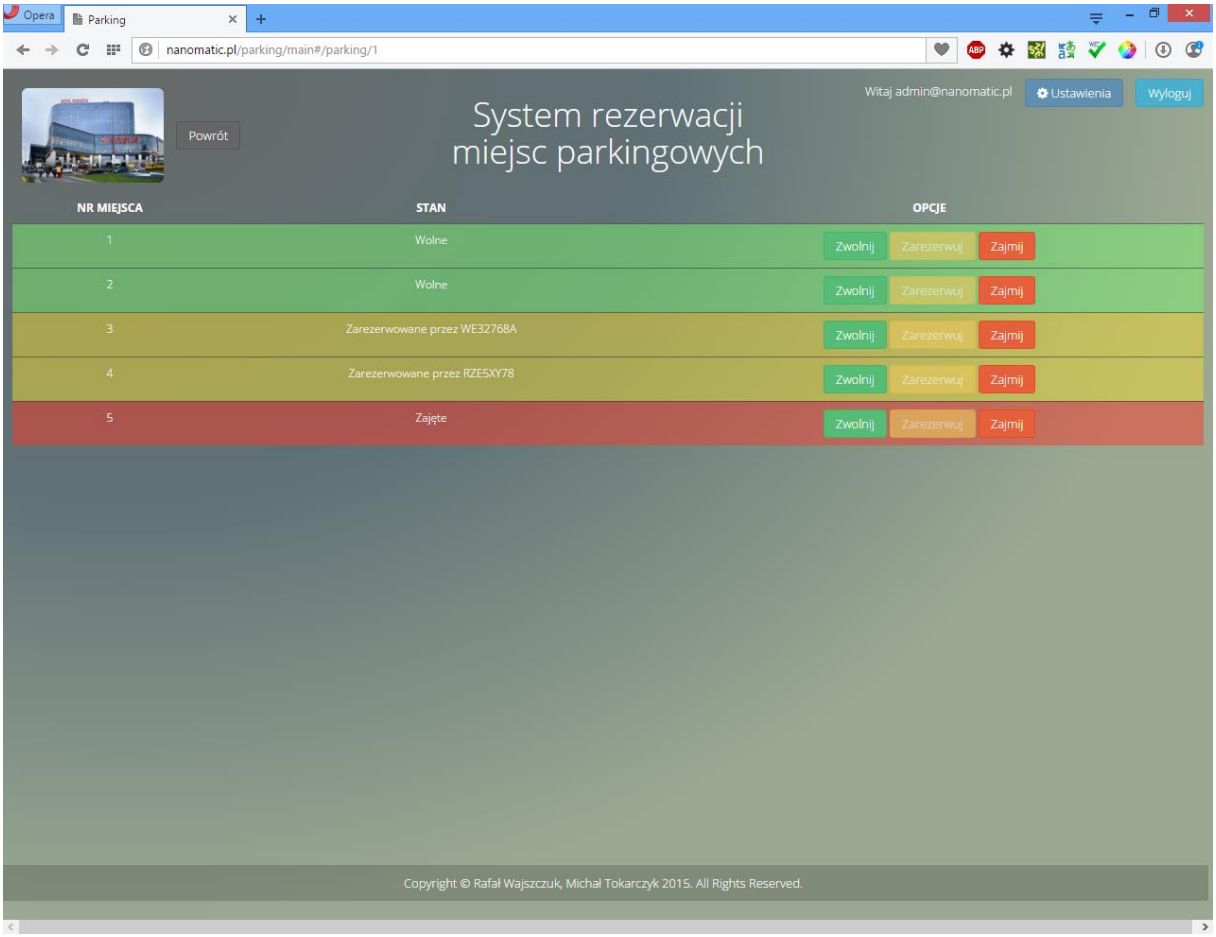
Rezerwacje mogę być wykonane jednokrotnie dla każdego z parkingów. W przypadku kiedy nastąpi zmiana stanu miejsca można wykonać ten proces ponownie. Jest to zabezpieczenie przed rezerwowaniem wszystkich miejsc przez jednego użytkownika.



rys. 16. Rezerwacja miejsca parkingowego na nr rej. RZE87K65

Administrator

Utworzone zostało specjalne konto administratora, celem możliwości zasymulowania zdarzeń występujących na realnych parkingach. Mając uprawnienia administratora (admin@nanomatic.pl / nanopass) możliwe jest zwalnianie, rezerwacja oraz zajmowanie miejsc na dowolnie wybranym parkingu. Wszystkie dokonane rezerwacje mają dodatkowo informacje o numerze rejestracyjnym na które zostały wykonane (rysunek 17).



rys. 17. Widok parkingu z uprawnieniami administratora

Niezależnie od uprawnień, system odpowiada w czasie rzeczywistym na zmiany dokonywane na parkingach, wobec czego zbędne staje się odświeżanie strony, aby uzyskać aktualny stan. Tę funkcjonalność można zaobserwować uruchamiając aplikację w dwóch odrębnych oknach lub na dwóch równych urządzeniach.