07.06.2015

**Autor**

Paweł Maciejewski

Podstawy teleinformatyki – dokumentacja

Scentralizowane zarządzanie routerami OpenWrt

2017

[1. Wstęp 4](#_Toc484701496)

[1.1. Temat 4](#_Toc484701497)

[1.2. Cel projektu 4](#_Toc484701498)

[1.3. Uzasadnienie wyboru 4](#_Toc484701499)

[2. Planowanie 5](#_Toc484701500)

[2.1. Planowana funkcjonalność 5](#_Toc484701501)

[2.2. Zrealizowana funkcjonalność 5](#_Toc484701502)

[2.3. Podział prac 6](#_Toc484701503)

[3. Praca wstępna 7](#_Toc484701504)

[3.1. Dostępne urządzenia 7](#_Toc484701505)

[3.2. Wybór technologii 7](#_Toc484701506)

[3.3. Uzasadnienie wybory danych technologii 7](#_Toc484701507)

[3.4. Usługi zewnętrze 8](#_Toc484701508)

[3.5. Składnia UCI 8](#_Toc484701509)

[3.6. Wysyłanie poleceń z uwzględnieniem składni UCI 8](#_Toc484701510)

[3.7. Klasa SshConnection i RoutersConnection 9](#_Toc484701511)

[3.8. Konfiguracja powiadomień 10](#_Toc484701512)

[3.9. Spiner podczas ładowania 11](#_Toc484701513)

[4. Napotkane problemy 12](#_Toc484701514)

[4.1. Problem z udostępnieniem routera oddalonego o 70 km. 12](#_Toc484701515)

[4.2. Problem z bazą danych w usłudze Amazon Web Services 12](#_Toc484701516)

[4.3. Problem z zespołem 13](#_Toc484701517)

[4.4. Pozostałe problemy 13](#_Toc484701518)

[5. Zarządzenie routerami 14](#_Toc484701519)

[5.1. Wyświetlanie dodanych routerów 14](#_Toc484701520)

[5.2. Dodawanie nowego routera 14](#_Toc484701521)

[5.3. Modyfikacja dodanego routera 15](#_Toc484701522)

[5.4. Usuwanie dodanego routera 15](#_Toc484701523)

[5.5. Ponowne połączenie wszystkich routerów 15](#_Toc484701524)

[5.6. Logowanie do aplikacji 16](#_Toc484701525)

[6. Wyświetlanie pełnej konfiguracji 17](#_Toc484701526)

[7. Wybór zapisanego routera 18](#_Toc484701527)

[8. Konfiguracja sieci WiFi 20](#_Toc484701528)

[9. Konfiguracja Firewalla 22](#_Toc484701529)

[10. Wyświetlanie podstawowych danych o połączeniu 25](#_Toc484701530)

[11. Podsumowanie 26](#_Toc484701531)

1. Wstęp

* 1. Temat

Projekt miał na celu stworzenie aplikacji pozwalającej na zarządzanie routerem opartym na systemie OpenWRT. OpenWRT jest oparty na podstawie systemu operacyjnego GNU/Linux stworzonego dla jednej z serii routerów Linksys przez ich producenta. Firma udostępniła go na licencji GNU GPL co oznacza, że musiała w całości opublikować kod. System ten został bardzo pozytywnie przyjęty przez społeczność i dzięki sporemu gronu entuzjastów doczekał się kolejnych wersji, m.in.: [DD- RT](https://pl.wikipedia.org/wiki/DD-WRT), [Tomato](https://pl.wikipedia.org/wiki/Tomato), [HyperWRT](https://pl.wikipedia.org/wiki/HyperWRT), [OpenWrt](https://pl.wikipedia.org/wiki/OpenWrt), Gargoyle czy FreeWRT.

* 1. Cel projektu

Głównym założeniem było stworzenie przyjaznego interfejsu użytkownika, dzięki któremu będzie on mógł wprowadzać zmiany w routerze nie mając specjalistycznej wiedzy z zakresu architektury Linuxa oraz poleceń i komunikatów obsługiwanych przez system. Projekt zakładał podstawową automatyzację, jednak z powodu odejścia członka zespołu, tempo tworzenia kodu zdecydowanie spadło. W ostatecznej wersji projektu zaimplementowana została większość funkcjonalności w całości konfigurowana manualnie przez użytkownika.

* 1. Uzasadnienie wyboru

Za wyborem tego tematu przemawiało już skromne doświadczenie w konfiguracji domowego routera opartego na tym systemie. Warto więc było poszerzyć tę wiedzę. Dodatkowo, obecnie na rynku nie istnieje podobne narzędzie, więc interesująca okazała się możliwość komercjalizacji projektu. Na chwilę obecną jednak jest to niemożliwe, gdyż z powodu problemów z zespołem projekt nie jest na tyle rozbudowany, aby mógł być sprzedawany. Jednakże z pewnością zostanie udostępniony na licencji Open Source.

1. Planowanie
   1. Planowana funkcjonalność

* Dodawanie wielu routerów, przechowywanie ich danych dostępowych w bazie danych oraz określanie ich dostępności w danym momencie
* Wyświetlenie pełnej konfiguracji danego routera
* Edycja danych sieci WiFi (nazwa, klucz, kanał itp.)
* Obsługa Firewall’a, czyli blokowanie klientom dostępu do sieci po ich adresie MAC lub IP oraz ruchu sieciowego z danego portu, na dany port, na dany adres IP
* Dodawanie i konfiguracja skryptów uruchamianych przy uruchomieniu systemu
* Podgląd aktualnych parametrów systemu (RAM, procesor)
* Proste monitorowanie ruchu sieciowego
* Automatyczna aktualizacja każdego z tych parametrów w routerze (o ile zostały one zmodyfikowane np. zewnętrznie)
  1. Zrealizowana funkcjonalność

Ze względu na odłączenie się od zespołu jednej osoby niemożliwe było zrealizowanie pełnej funkcjonalności. Jednakże zrealizowano jej ważniejszą część udostępniając ładnie prezentującą się aplikację z podstawową funkcjonalnością. Stopień realizacji założonych celów na 2 osoby można uczciwie określić na 70% (praca tylko jednej osoby). Oznacza to, że jeden autor wykonał 140% pracy, która mu powinna przypadać.

Zrealizowano:

* Całą pracę wstępną, czyli wybór technologii, konfigurację projektu, analizę sposobu komunikacji, przygotowanie bazy danych, zabezpieczenie dostępu do aplikacji loginem i hasłem, zaimplementowanie wzorców projektowych
* 2 z 3 zadań trudnych – zrealizowano dodawanie routerów i przełączanie pomiędzy nimi i dość rozbudowaną opcję Firewall’a, nie zrealizowano automatycznej aktualizacji parametrów w routerze wraz z watchdog’iem monitorującym ,czy ustawienia nie zostały zmodyfikowane zewnętrznie
* 3 z 5 zadań łatwych – zrealizowano wyświetlanie pełnej konfiguracji routera, edycję sieci danych sieci WiFi oraz wyświetlanie parametrów systemu, nie zrealizowano dodawania i konfiguracji skryptów uruchamianych przy starcie systemu i prostego monitorowania ruchu sieciowego
* Frontend z intuicyjnym interfejsem użytkownika opartym na Bootstrapie
  1. Podział prac

Wstępny podział prac zakładał współpracę dwóch członków, ale z powodu rezygnacji jednego z członków z uczelni całość została wykonanana przez jedną osobę.

1. Praca wstępna
   1. Dostępne urządzenia

* Tp-Link WR1043ND
* TP-Link TL-WR740N
* Serwer zdalny z systemem Ubuntu do zdalnego podłączenia jednego z routerów
  1. Wybór technologii
* ASP .NET MVC
* IDE: Visual Studio 2015
* SSH do łączenia z routerami, wykorzystano bibliotekę SSH.NET
* Wzorce projektowe: MVC, Repository Pattern wraz z Uit of Work
* Baza danych: Microsoft SQL Serwer
* Bootstrap, Razor i JS jako frontend
  1. Uzasadnienie wybory danych technologii
* Dobra znajomość technologii Microsoftu i praca zarobkowa z ich użyciem, dlatego wykorzystano technologię webową ASP .NET MVC, środowisko deweloperskie Visual Studio 2015 oraz bazę danych w standardzie Microsoft SQL
* Zastosowano wzorce projektowe, aby ułatwić pisanie kodu i poprawić jego czytelność. Wzorzec MVC był konieczny, bowiem jest zintegrowany z technologią ASP .NET MVC. Wzorzec repozytoriów jest chyba najprostszym wzorcem do łatwej obsługi baz danych obsługiwanych za pomocą narzędzia ORM Entity Framework. Dodanie do niego Unit Of Work’a pozwala na umieszczenie bardziej rozbudowanej logiki bazy danych w wykodny i dynamiczny sposób.
* SSH ponieważ nie da się prościej skomunikować z systemem OpenWRT. Inną możliwością byłoby wykorzystanie gotowego interfejsu graficznego i jego analiza lub połączenia szeregowego. Nawiązanie połączenia przez SSH jest więc najprostrze i najbardziej funkcjonalne, gdyż daje dostęp do np. składni UCI.
* Bootstrap ponieważ wiele rozwiązń jest już zaimplementowanych
* JavaScript ponieważ część logiki typu sortowanie wyników musiało być wykonane po stronie klienta
* Razor ponieważ jest zintegrowany z ASP .NET MVC
  1. Usługi zewnętrze
* Baza danych w Amazon Web Services
* Serwer Ubuntu w Digital Ocean
* Repozytorium w serwisie GitHub
  1. Składnia UCI

Interfejs "uci" (Unified Configuration Interface - ujednolicony interfejs konfiguracyjny) jest sposobem konfiguracji systemu stosowanym w OpenWrt. W przeciwieństwie do stosowanego w starych wydaniach nvram też inne zalety - mniejsze zużycie flash (nie jest ciągle zapisywany ten sam obszar, tylko przydzielane jest losowo miejsce do zapisania pliku - wynika z właściwości systemu plików) czy możliwość zastosowania go w normalnych systemach plików (np. dla x86).

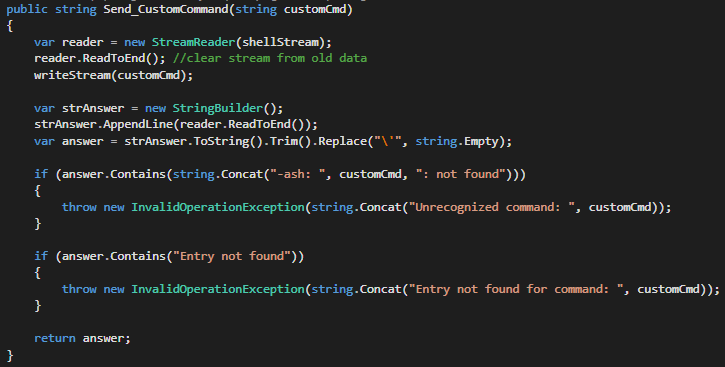
Domyślnie pliki konfiguracyjne zawarte są w katalogu "/etc/config”. Ważna jest nazwa pliku (przez nią odwołujemy się do poszczególnych sekcji) oraz jego budowa w postaci wydzielonych sekcji.

Odwołanie zawsze następuje wg schematu:

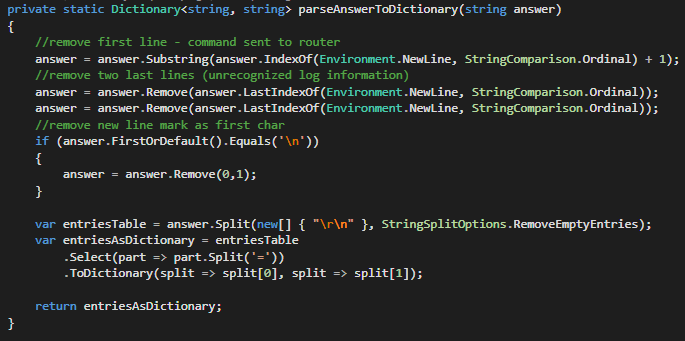
konfig.sekcja.nazwa\_opcji=wartość

* 1. Wysyłanie poleceń z uwzględnieniem składni UCI

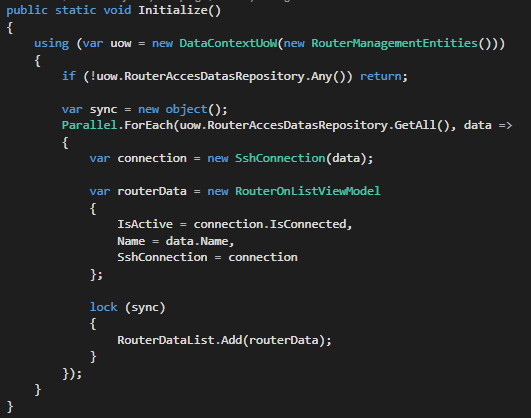
Polecenie jest wysyłane do streamu writera, a odczyt następuje ze streamu readera. Następnie następuje sprawdzenie, czy odpoweidź nie zawiera komunikatu błędu.



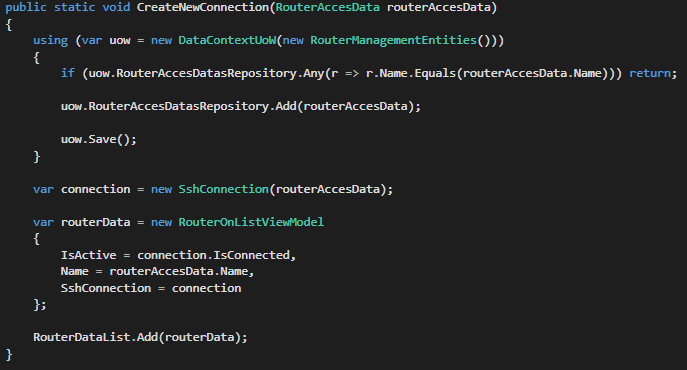
Następnie odpowiedź jest parsowana do słownika, zawierającego nazwę klucza wraz z sekcją i jego wartość.



* 1. Klasa SshConnection i RoutersConnection
* SshConnection zawiera niskopoziomowe metody do zarządzania routerem – dodawanie, odczyt i edycja ustawień oraz metody parsujące wynik na postać łatwą do obsługi w logice aplikacji (postać obiektowa). Ważniejsze funkcjonalności będą omówione w dalszej części dokumentacji.
* RoutersConnection zawiera listę routerów i metody pobierające routery i dane o nich wg. Kryteriów, np. pobieranie aktywnych routerów czy pobieranie samych nazw routerów. Każdy z obiektów routerów zawiera własną klase SShConnection, dzięki czemu połączenie z każdym z nich odbywa się niezależnie.
* Inicjalizacja routerów odbywa się poprzez odczyt z bazy danych za pomocą repozytorium danych o nich. Następnie wielowątkowo odbywa się ich inicjalizacja w tle.

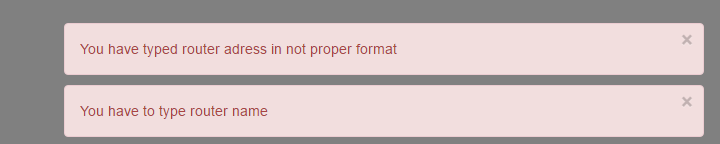


* Dodawanie routera do aplikacji składa się z dodania danych dostępowych do bazy danych oraz dodania go do listy routerów z nawiązanych połączeniem (lub też nie – połączenie może nawiązać użytkownik później ręcznie



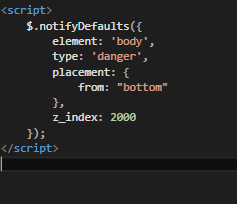
* 1. Konfiguracja powiadomień

Gdy zajdzie potrzeba, użytkownik musi być powiadomiony o danej akcji. Np. gdy podał niepoprawne dane lub ich nie podał przy konfiguracji jednej z funkcji. Powiadomienia sa wyświetlane w prawym dolnym rogu ekranu w następujący sposób:



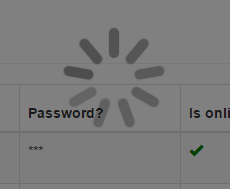
Do tego celu jest korzystywany plugin „bootstrap-notify”, który wyświetla powiadomienia w stylu Bootstrapa, dzięki czemu style w całej aplikacji są jednolite.

Konfiguracja tego wygląda następująco:

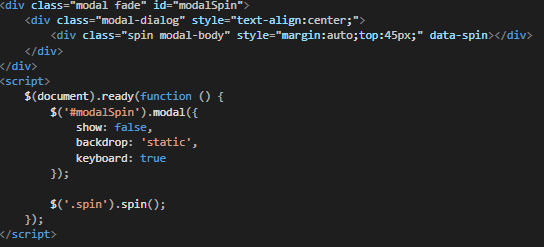


* 1. Spiner podczas ładowania

Aby użytkownik nie musiał oglądać strony wczytujacej się nieraz dłuższą chwilę (restart serwisów może chwilę trwać) zostaje wyswietlony spinner.



Wykorzystano gotowy plugin „jquery.spin”. Jest on dodany na każdej stronie jako partial i zależnie od potrzeby, wyswietlany bądź chowany. Konfiguracja wygląda następująco:



1. Napotkane problemy
   1. Problem z udostępnieniem routera oddalonego o 70 km.

System miał za zadanie zarządzać kilkoma routerami, w związku z czym podjęto próbę udostępnienia zdalnie drugiego routera. Otworzenie portu 22 na zewnątrz nie było problemem, problemem okazał się operator Play. Nie udostępnia on stałego IP, ale bez problemu można ten problem rozwiązać usługą dynamicznego DNS, do którego router loguje się co określoną ilość czasu i aktualizuje swój obecny adres IP na serwerze, a w zamian otrzymuje adres mnemoniczny. Poważniejszym problemem okazała się natomiast blokada połączeń przychodzących. Taka blokada jest nakładana dla bezpieczeństwa użytkowników sieci mobilnych, aby złośliwy serwer z zewnątrz nie naraził użytkownika na opłaty z tytułu opłaty za transfer. Operator odmówił również zdjęcia tej blokady po prośbie w zgłoszeniu do konsultanta.

 (…) dziękuję za przesłane przez Pana zgłoszenie. Chciałbym wyjaśnić, że ze względów bezpieczeństwa, zablokowaliśmy porty dla ruchu przychodzącego. Konsultowałem Pana prośbę z naszym działem technicznym, bardzo mi przykro ale nawet na Pana prośbę, nie jesteśmy w stanie ich odblokować. (…)

Jako rozwiązanie zdecydowano się wykorzystać opcję Reverse-SSH, dzięki której połączenie nie jest nawiązywane bezpośrednio do routera, tylko do zdalnego serwera, który przekazuje dane do routera. Serwer nie łączy się z routerem, tylko router z serwerem, a połączenie jest cały czas podtrzymywane, dzięki czemu omija się blokadę połączeń przychodzących. Serwer utworzono w usłudze Digital Ocean wykorzystując darmowe 50$ otrzymane za fakt bycia studentem. Usługa działała, ale był problem z podtrzymaniem połączenia. W związku z brakiem czasu na znalezienie rozwiązania koncepcję porzucono.

* 1. Problem z bazą danych w usłudze Amazon Web Services

Do realizacji projektu miała zostać użyta baza danych zdalna. Znalezienie bezpłatnego serwera baz danych SQL Server było trudne. Darmowy serwer Microsoft Azure dla studentów był już użyty w innym projekcie. Rozwiązaniem okazał się bezpłatna oferta w serwisie AWS o dość dobrych parametrach zlokalizowana w USA. Pojawiały się jednak problemy z nawiązaniem połączenia i prędkością transmisji. Ostatecznie zrezygnowano ze zdalnego serwera na rzecz bazy lokalnej, ponieważ zdalny serwer miał sens wówczas, gdy w projekcie przewidywane były dwie osoby.

* 1. Problem z zespołem

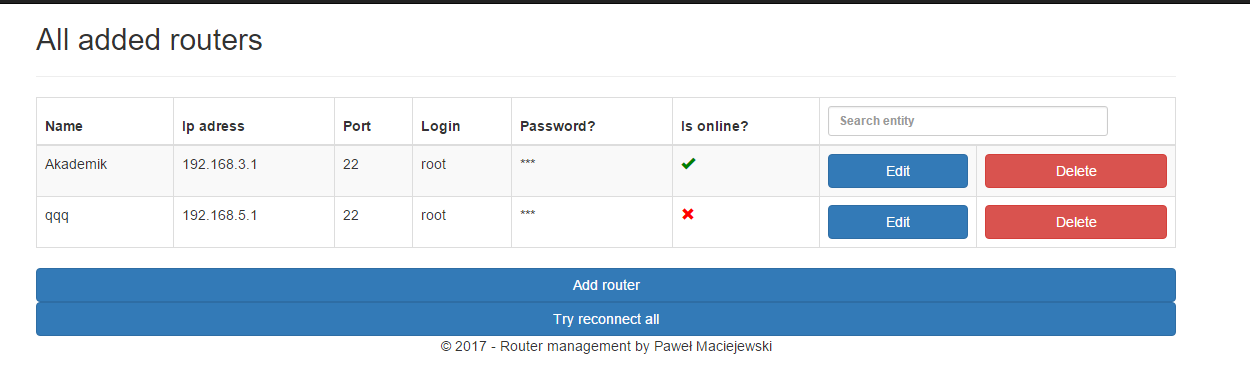
Nieoczekiwanie ze studiów zrezygnował członek zespołu. W związku z czym niemożliwe okazało się zrealizowanie pełnej planowanej funkcjonalności.

* 1. Pozostałe problemy

Pozostałe problemy związane z poszczególnymi funkcjonalnościami zostały poruszone w odrębnych sekcjach przeznaczonych dla nich.

1. Zarządzenie routerami
   1. Wyświetlanie dodanych routerów

Routery są przechowywane na liście,która zawiera dane kazdego routera, połączenie z nim oraz dane na temat tego połączenia. Połączenia sa wyswietlane w intuicyjnej formie w panelu uzytkownika.



* 1. Dodawanie nowego routera

Algorytm dodawania routera do bazy danych jest dość prosty. Uzytkownik po wciśnięciu przycisku dodawania routera widzi modal, w którym może te dane dodać.

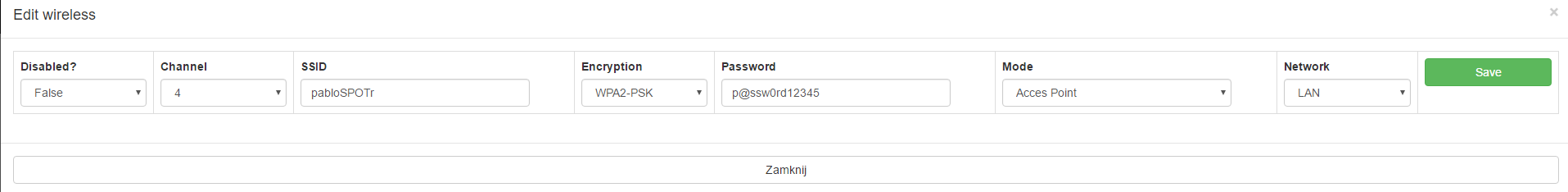
Po wpisaniu danych i dodaniu js analizuje je za pomoca wyrażen regularnych, a w przypadku braku z godności z wzorcem wyswietla komunikat błędu.



Jezeli są poprawne, przesyła dane do logiki aplikacji. Logika ponownie waliduje te dane – jest to zabezpiecznie przeciwko użyciem programów typu PostMan, które by pozowlily na wysłanie dowolnych danych do routera. Logika wpierw pinguje adres, aby nie marnowac czasu na nawiązanie połączenia z ewentualnym błędnym adresem.

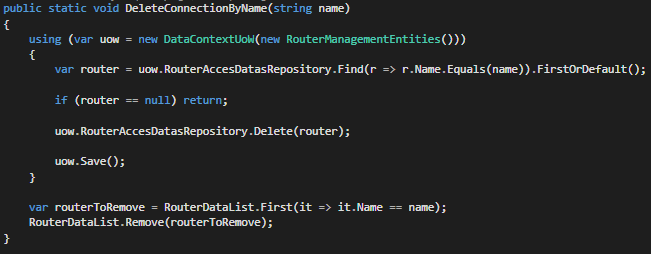
* 1. Modyfikacja dodanego routera

Po wciśnięciu przycisku modyfikacji dane na temat routera sa pobierane z bazy danych. Następnie wyswietlany jest ten sam modal, co w przypadku dodawania routera, ale tym razem z domyslnie podanymi wartościami. Wciśnięcie przycisku zapisu zmodyfikowanych danych powoduje uruchomienie tego samego procesu walidacji, co dla dodawania routera. Po modyfikacji danych routera w bazie danych nastepuje proces ponownego połączenia z routerem.



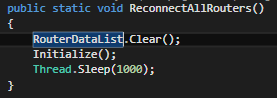
* 1. Usuwanie dodanego routera

Usunięcie powoduje skasowanie danych routera z bazy danych oraz z listy przechowywującej połączone routery. Następuje zwolnienie zasobów i zamkniecie streamów SSH.



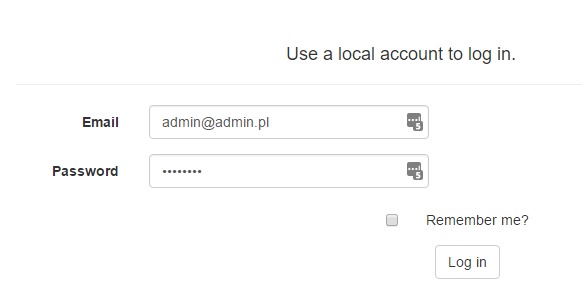
* 1. Ponowne połączenie wszystkich routerów

Funkcja, która miała w zamysle byc inicjowana co jakis czas, aby miec pewność, że wszystkie routery sa on-line. Z powodu braku tej funkcjonalności funkcja jest wyzwalana ręcznie z panelu administratora.



* 1. Logowanie do aplikacji

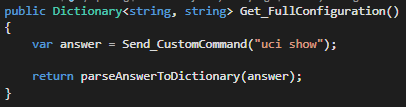
Wykorzystano gotową funkcjonalność logowania generowana przy tworzeniu projektu. Nie ma ona żadnej funkcjonalności typu zapomniane hasło, potwierdzenie e-maila. W projekcie skupiono sie na praktycznej funknjonalności i interfejsie uzytkownika.



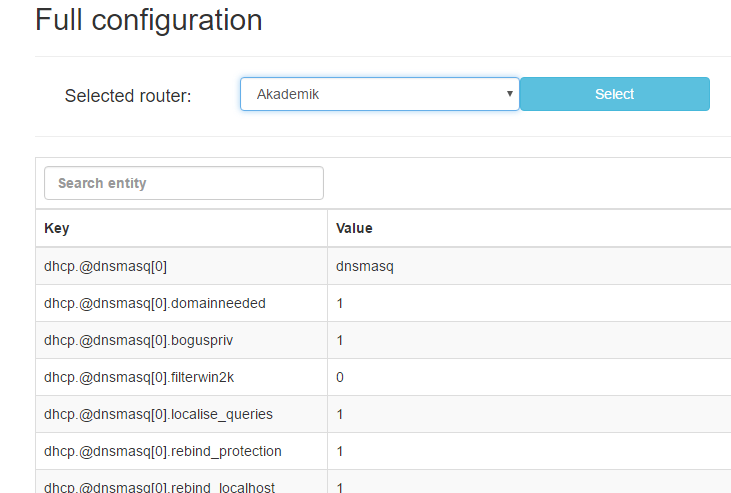
1. Wyświetlanie pełnej konfiguracji

Wykorzystano polecenie „uci show” wyświetlającą pełną konfigurację

Funkcja wysyłająca polecenie UCI Show wykorzystuje zaimplemntowany już wrapper SSH.

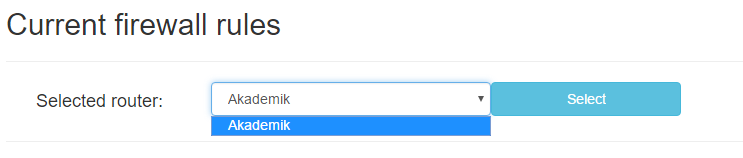


Dane w postaci słownika sa przesyłane do widoku, gdzie połączenie silnika Razor z plugin JavaScript’owym ListJS umożliwia wyświetlenie danych w przystępnej formie z jednoczesną możliwością sortowania wyników.

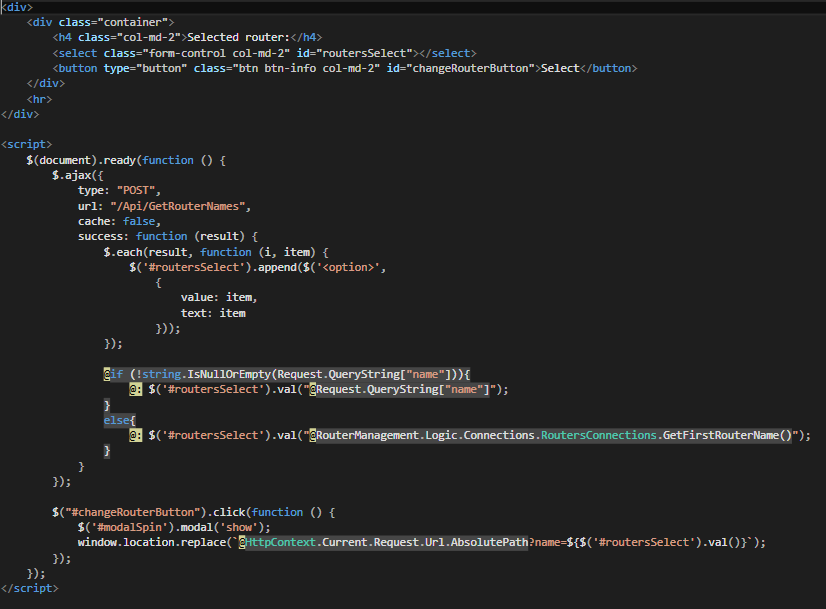


1. Wybór zapisanego routera

Serwis miał za zadanie zarządzać więcej niz jednym routerem w sposób zautomatyzowany oraz ręczny. Zapimplementowana została funkcja ręcznego zarządzania. Dla kazdej funkcjonalności mozliwe jest wybranie routera, dla którego ta funkcjonalność będzie realizowana. Poniżej przykład wyboru jednego z routerów on-line.

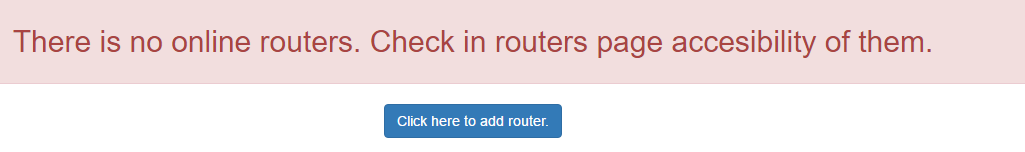


Okno wyboru routera jest partialem, który za pomocą js i silnika razor generuje listę i umieszcza ją w odpoweidnim miejscu. Przy wyborze odpowiedniej funkcji edytowany jest adres URL dodając do niego nazwe routera. Jeżeli taki parametr nie jest podawany, wyświetlany jest zawsze pierwszy router z listy.



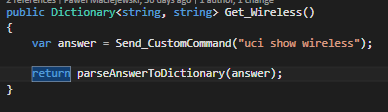
Dlatego ważne jest, aby routery nie miały nazwy zawierajacej znaki wykorzystywane w adresach URL. Dba o to wyrażenie regularne przy dodawaniu/edycji routera.

W momencie, gdy żaden router zapisany w bazie danych nie jest on-line, wysiwetlany jest stosowny komunikat.

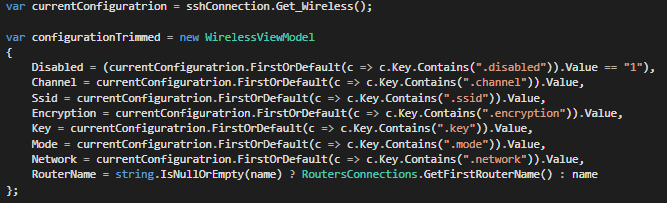


1. Konfiguracja sieci WiFi

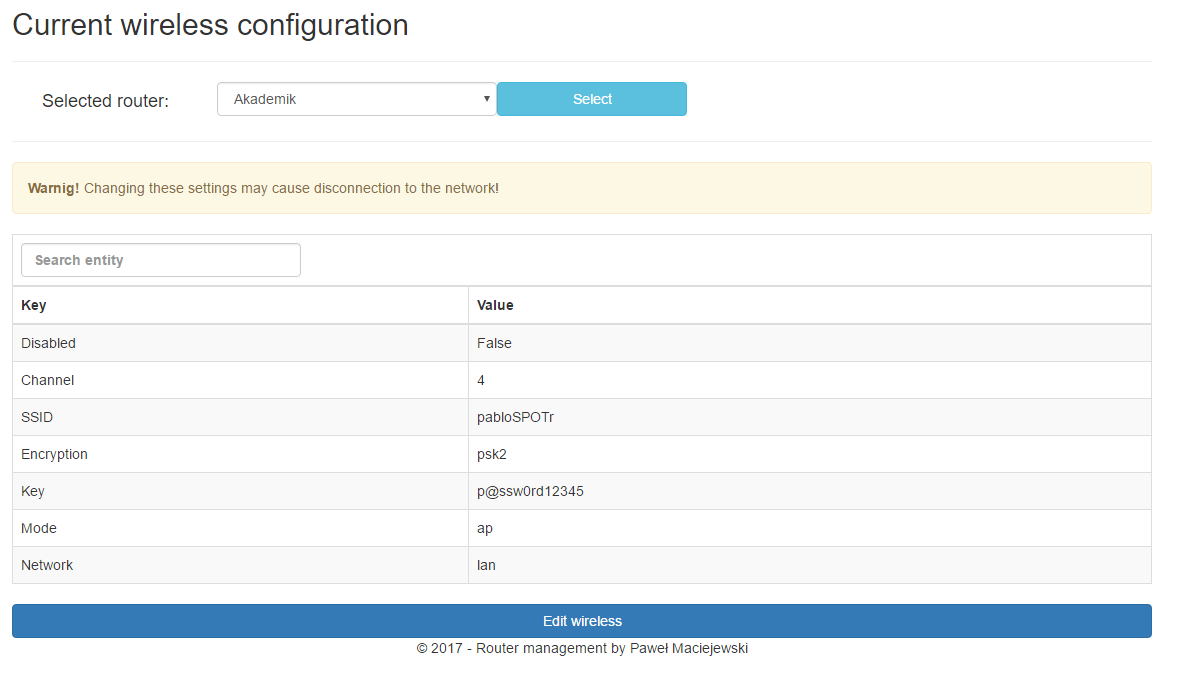
Przy wejściu w odpowiednią sekcję, dane na temat sieci wifi wybranego routera sa pobierane za pomocą gotowych metod i przepuszczane przez wrapper. Dlatego metoda ta jest bardzo prosta.



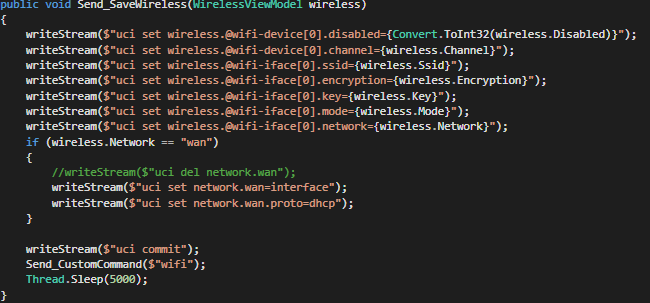
W nastepnym ktoku po prostu wybierane są wymagane potrzebne dane z sekcji „wireless”.



Wyświetlanie danych o sieci wygląda następująco:



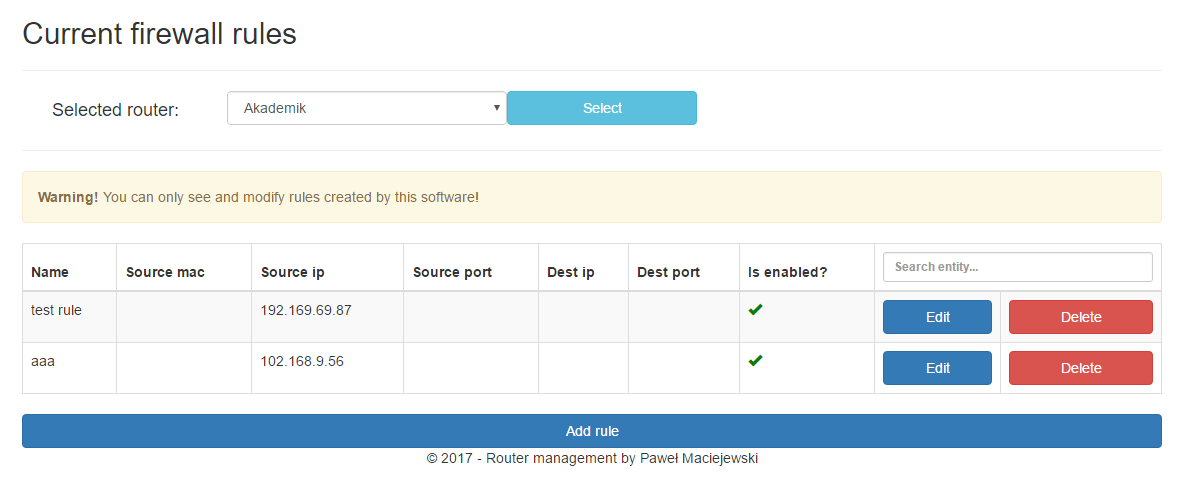
Sieć wifi można także zedutować. Poprawności podawanych danych pilnują wyrażenia. Zapis ustawien wygląda nastepująco:



Zmiana ustawień sieci WiFi może spowodować rozłączenie użytkownika, jeżeli jest on połączony bezprzewodowo, o czym jest poinformowany.

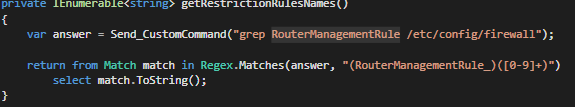
1. Konfiguracja Firewalla

Funkcjonalność firewalla pozwala na wykonywanie podstawowych operacji, tj. blokowanie dostępu do sieci dla użytkownika o danym adresie MAC, IP, ruchu z danego portu, na dany port i na dany adres. Reguła może byc włączona lub wyłączona. Wszystkie reguły przechowywane są na danych urządzeniach.

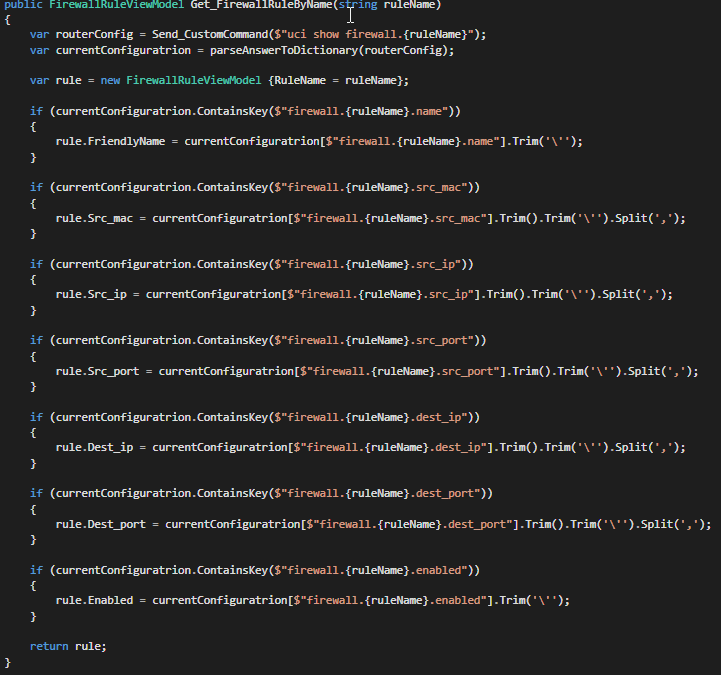


Podstawowym problemem obsługi firewalla było to, że w systemie istnieje juz kilka standardowych reguł w dość zaawansowanej postaci. Tworzenie obsługi wszystkich funkcjonalności przez nie obsługiwanych zajęłoby zdecydowanie zbyt dużo czasu, w związku z czym konieczne było znalezienie sposobu, aby obsługiwać tylko reguły tworzone przez dane oprogramowanie. Rozwiązaniem okazała się funkcjonalność UCI, która pozwala nadawać danym sekcjom ich dodatkową nazwę. Dzięki czemu każda z reguł dostała unikatową nazwę „RouterManagementRule\_X” (gdzie X to jej unikalny numer).

Pobieranie reguł polega na odczycie ich nazw, a potem odczycie ustawień dla danej nazwy. Nazwy są wydobywane z pliku /etc/config/firewall za pomocą funkcji grep. Nastepnie wyrażenie regularne odczytuje wszystkie wyrazy spełniające wzór – czyli nazwy reguł.



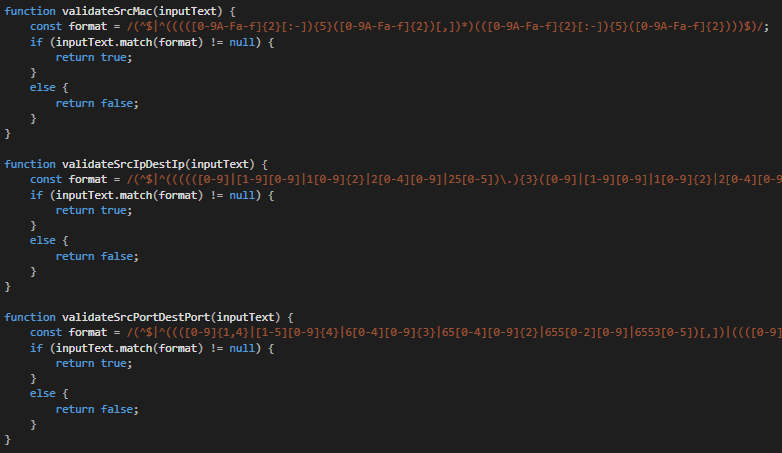
Nastepnie każda z reguł jest pobierana z routera.



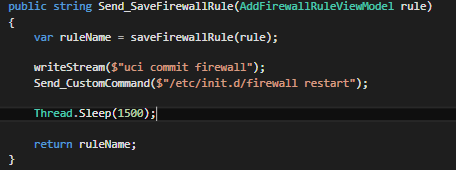
System pozwala także na dodawanie reguł.

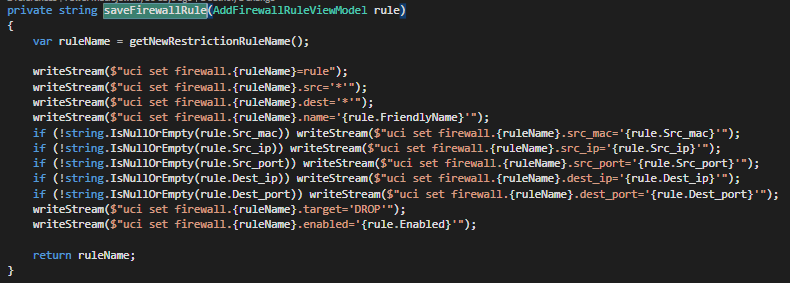


Nad poprawnością danych, równiez jak w innych modułach, stoją wyrażenia regularne.

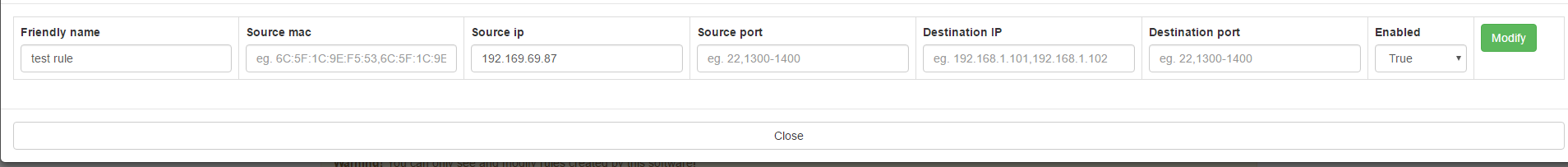


Zapis wygląda następująco:





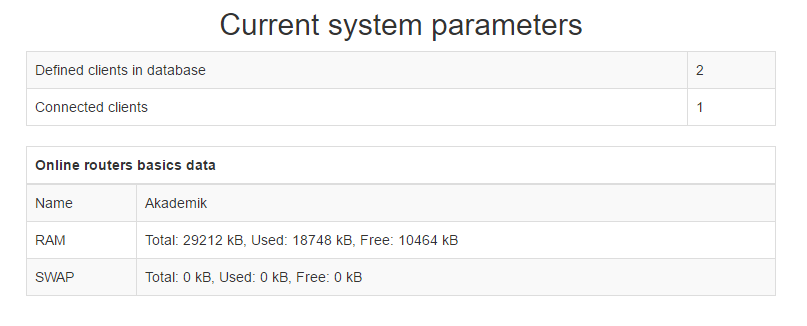
Reguły można także edytować.



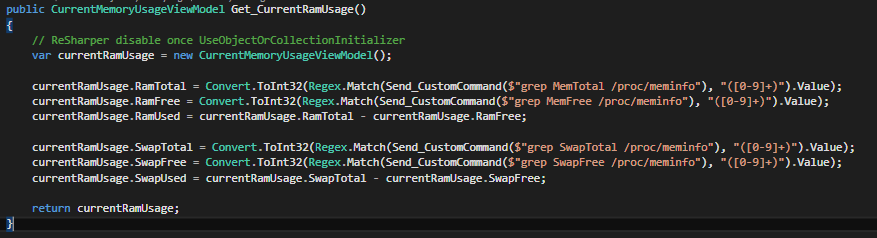
1. Wyświetlanie podstawowych danych o połączeniu

Aplikacja na głównej stronie panelu admina wyswietla podstawowe dane o połączonych routerach.

* Ilość zapisanych routerów i ilość routerów online
* Pamięć RAM i SWAP oraz jej zuzycie



Pobieranie informacji o pamięci polega na odczycie pliku /proc/meminfo.



1. Podsumowanie

* Nie wykorzystano żadnego systemu pracy, bowiem nad projektem pracowała jedna osoba
* Na poczatku byly pisane testy jednostkowe, ale ze względu na małą ilość czasu porzucono te koncepcję (niestety)
* Realizacja projektu pozwoliła nauczyć mi sie wielu ciekawych rozwiązań
* Użycie Githuba ułatwiło mi pracę na różnych komputerach
* Praktyczne wykorzystanie wzorców projektowych z pewnością zaowocuje w pracy programisty