Politechnika Wrocławska
Wydział Elektroniki

---

Kierunek: Informatyka
Specjalność Inżynieria Internetowa

# Projekt Inżynierski

Internetowy system wspomagania treningów aerobowych z aplikacją mobilną

Internet system for aerobic training with a mobile application

Autor:

Jacek Wieczorek

Prowadzący pracę:

dr inż. Tomasz Walkowiak

Ocena pracy:

---

Wrocław, 2012

# Spis treści

# Rozdział 1

# Wprowadzenie

## 1.1. Wstęp

Popularyzacja sportu wśród społeczeństwa jest jednym z głównych celów dzisiejszych kampanii na rzecz zdrowego stylu życia.

## 1.2. Cel pracy

Celem niniejszego projektu inżynierskiego jest zaprojektowanie oraz zaimplementowanie systemu do wspomagania treningów areobowych. Główną częścią systemu jest aplikacja internetowa wraz z aplikację mobilną. By zapewnić możliwość przesyłania danych do serwera, konieczne jest stworzenie API, będącego pośrednikiem między aplikacją mobilną, a bazą danych. System ma pozwolić użytkownikom zarządzać treningami, prezentować swoje osiągnięcia innym osobom korzystającym z portalu i swobodnie komunikować się z nimi.

Podstawowe wymagania, które ma spełniać projekt:

- tworzenie i edycja treningów,
- tworzenie i edycja profili użytkowników,
- komunikacja pomiędzy użytkownikami,
- synchronizacja danych z aplikacji mobilnej,
- zarządzanie prywatnością profilu użytkownika,
- możliwość przeglądania treningów innych użytkowników,
- stworzenie formy małego portalu społecznościowego.

Zakres prac wykoannych w projekcie:

- projekt systemu - rozdział 2
- implementacja aplikacji internetowej - rozdział 3
- implementacja aplikacji mobilnej - rozdział 4
- implementacja REST Api - rozdział 5

## 1.3. Istniejące rozwiązania

Na rynku dostępnych jest wiele rozwiązań pozwalających na zarządzanie treningami np. *Endomondo* lub *Nike Running*. Jednak większość z nich jest aplikacjami komercyjnymi, których darmowe produkty obarczone są dużą ilością reklam lub ograniczoną liczbą dostępnych funkcjonalności. Ponadto aplikacje te często zatracają swoją idę wspierania treningów i stają się kolejnymi portalami typu Social Media.

Celem tego projektu jest stworzenie systemu, oferującego podobne możliwości jak oferowane na rynku rozwiązania, nie tracąc przy tym na prostocie użytkowania i przejrzystości interfejsu użytkownika.

# Rozdział 2

# Opis systemu i komponenty wykorzystywane w projekcie

## 2.1. Projket systemu

Zaprojektowany system składa się z 4 kluczowych elementów

## 2.2. C# i .NET

C# jest obiektowym językiem programowania stworzonym dla firmy Microsoft przez zespół Andersa Hejslberga. Język ten powstał jako alternatywa dla Javy. Programy napisane w C# kompilowane są do natywnego języka Common Intermediate Language (CIL), czyli kodu pośredniego w środowisku takim jak .NET czy Mono.

.NET Framework jest platformą programistyczną obejmującą środowisko uruchomieniowe CLR (Common Language Runtime) oraz biblioteki klas z podstawowymi funkcjonalnościami. Zadaniem .NET jest zarządzanie elementami systemu, takimi jak: kod aplikacji, pamięć oraz zabezpieczenia. .NET umożliwia uruchamianie aplikacji zarówno na systemie, w którym istnieje działająca implementacja platformy oraz po stronie serwera IIS. Platforma .NET nie określa jednoznacznie języka programowania. Aplikacje działające przy wykorzystaniu platformy .NET mogą być napisane w C#, F#, J#, VB.NET, C++/CLI czy Delphi 8.

## 2.3. Framework ASP.NET MVC 3

ASP.NET MVC 3 jest platformą do budowy aplikacji internetowych korzystającą z wzorca MVC (Model - Widok - Kontroler) bazującej na platformie ASP.NET. Zaletą z korzystania ze

wzorca MVC jest odseparowanie warstw aplikacji i logiki biznesowej. Aplikacje stworzone za pomocą tego frameworka są z reguły łatwiejsze w rozbudowie i testowaniu (testy jednostkowe).

ASP.NET MVC bazuje na tradycyjnym silniku ASP.NET, dzięki czemu można wykorzystać wiele mechanizmów stworzonych dla tej platformy jak zarządzanie cachem, autoryzacja czy monitorowanie. Mechanizm mapowania adresów umożliwia łatwą budowę aplikacji w oparciu o architekturę REST. Model programistyczny silnie bazuje na interfejsach, co pozwala na łatwą rozbudowę i testowanie komponentów.

## 2.4. Windows Communication Foundation

Windows Communication Foundation (w skrócie WCF) jest platformą służącą do budowy aplikacji zorientowanych serwisowo. Pozwala na budowę serwisów, które mogą działać na serwerze IIS lub jako część systemu. WCF pozwala na komunikację między platformową za pomocą technologii SOAP przy użyciu prostych formatów XML lub JSON. Kluczowym aspektem biznesowym platformy WCF jest zapewnienie wysokiej wydajności, przy równie wysokiej niezawodności działania systemu.

## 2.5. Baza danych

### 2.5.1. Microsoft SQL Server 2012

Microsoft SQL Server (MS SQL) to system zarządzania relacyjnymi bazami danych. Jako język zapytań używany jest Transact-SQL będący rozwinięciem standardu ANSI/ISO. MS SQL jest platformą bazodanową typu klient-serwer charakteryzującą się wysoką wydajnością, niezawodnością i bezpieczeństwem.

### 2.5.2. SQLite

SQLite jest systemem zarządzania bazą danych i biblioteką języka C, implementującą taki system. Charakteryzuje się przetrzymywaniem danych w jednym pliku (do 2 TB). Baza utrzymywana jest przy użyciu B-drzewa. Bazy danych zapisywane są jako pliki binarne. SQLite jest szeroko wykorzystywany w systemach wbudowanych jak i przez platformę Android.

## 2.6. Windows Azure

Windows Azure jest platformą chmurową stworzoną przez firmę Microsoft. Udostępnia ona mechanizmy do przetwarzania danych (Windows Azure Compute) oraz do ich przechowywania (Windows Azure Storage i SQL Azure).

Windows Azure pozwala na budowanie aplikacji we wszystkich technologiach, które można wykorzystać na zwykłym systemie Windows. Poza C# i innymi językami platformy .NET można tworzyć oprogramowanie w takich językach jak Java, C++, PHP czy Python. Windows Azure zapewnia integracją z popularnymi środowiskami programistycznymi jak Microsoft Visual Studio czy Eclipse.

W celu ułatwienia procesu wytwarzania oprogramowania pod tę platformę, firma Microsoft udostępniła dedykowane dla różnych języków zestawy narządzi programistycznych. Ważną ich częścią jest emulator chmury, dzięki któremu można w wygodny sposób testować działanie aplikacji, bez konieczności instalacji oprogramowania na serwerze.

## 2.7. Android SDK

Android SDK jest paczką narzędzi programistycznych, niezbędnych w procesie tworzenia i testowania aplikacji na platformę Android. Ważnymi elementami Android SDK są wtyczki dla środowiska programistycznego Eclipse i symulator systemu.

## 2.8. CoffeeScript

CoffeeScript jest językiem skryptowym kompilowalnym do kodu JavaScript. Główną zaletą tego języka jest wyeliminowanie nawiasów, klamr i średników, co zmniejsza ryzyko popełnienia błędów składniowych, uniemożliwiających poprawne działanie aplikacji internetowej.

## 2.9. HTML5 i CSS3

HTML5 jest językiem wykorzystywanym do tworzenia stron WWW. Jest rozwinięciem standardu języka HTML4, posiadając wiele udogodnień dla programistów. Głównymi technologie wersji 5 są obsługa grafiki 2D i 3D, audio i video oraz pełne wsparcie dla kaskadowych arkuszów styli w wersji 3 (CSS3). Dzięki temu, nie trzeba posiłkować się bibliotekami JavaScript oraz technologią Flash, by osiągnąć oczekiwane efekty wizualne i użytkowe aplikacji.

## 2.10. Inne biblioteki wykorzystane w projekcie

### 2.10.1. Entity Framework

Entity Framework jest biblioteką służącą do relacyjno-obiektowego mapowania w środowisku ADO.NET. Dzięki zastosowaniu tej biblioteki wraz z narzędziami będącymi integralną częścią języka C#, można w wygodny sposób zarządzać bazą danych bez konieczności używania języka SQL.

### 2.10.2. AutoMapper

AutoMapper jest biblioteką służącą do mapawania obiektów jednego typu na drugi. Pozwala to w szybki sposób odzielić warstwę aplikacji od warstwy biznesowej na poziomie modelu aplikacji.

### 2.10.3. Unity

Unity jest lekkim, łatwo rozszerzalnym kontenerem do wstrzykiwania zależności, działającym na platformie .NET. Umożliwia ono programistom osiągnięcie następujących korzyści:

- uproszczenie tworzenia obiektów,
- uwiększenie elastyczności poprzez konfigurację komponentów w kontenerze,
- możliwość lokowania serwisów.

# Rozdział 3

# Aplikacja internetowa

## 3.1. Architektura aplikacji

### 3.1.1. Wzorzec Model-Widok-Kontroler

Model-Widok-Kontroler (*ang. Model-View-Controller*) w skrócie MVC, jest wzorcem projektowym rozdzielającym aplikację internetową na 3 warstwy: model, widok i kontroler, które komunikują się ze sobą wzajemnie (Rysunek 3.1).

#### Model

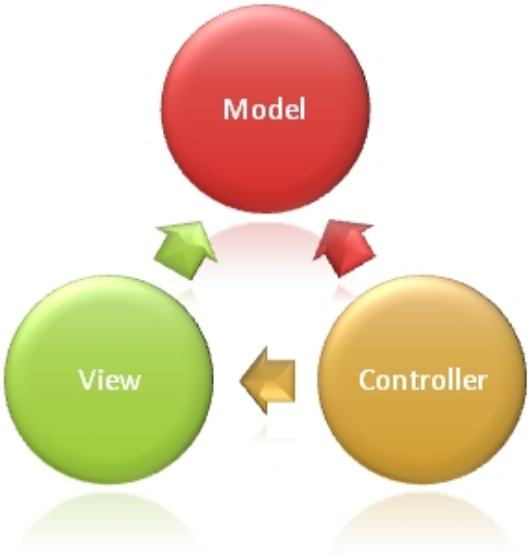
Warstwa modelu odpowiada za reprezentację logiki systemu oraz dostęp do bazy danych. W projekcie za tę część aplikacji odpowiadają dwie biblioteki DLL : **PI.Service** oraz **PI.Data**.

#### Widok

Widok jest warstwą odpowiedzialną za wyświetlanie interfejsu użytkownika. Najczęściej widoki generowane są na podstawie modelu. W apliakcjach MVC widoki tylko wyświetlają informacje. Ta warstwa znajduje się w bibliotece **PI.Web**.

#### Kontroler

Kontrolery to komponenty odpowiedzialne za utrzymanie interakcji z użytkownikiem, pracę z modelem i renderowaniem odpowiednich widoków. Kontrolery obsługują rządania użytkowników, konwertują parametry zapytań na modele i przekazują je do kolejnej warstwy. Kontrolery, podobnie jak widoki znajdują się w bibliotece **PI.Web**.




Rys. 3.1: Schemat graficzny wzorca Model-Widok-Kontroler

### 3.1.2. Wzorzec Repozytorium

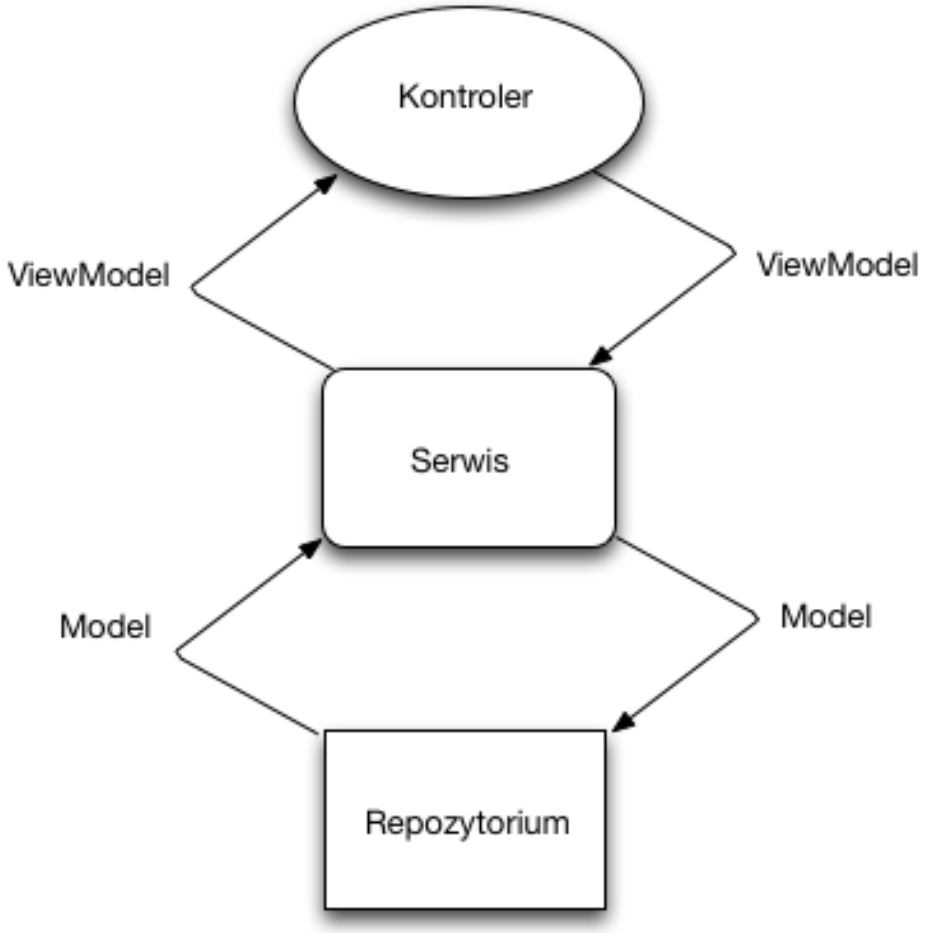
Wzorzec Repozytorium (*Repository Pattern*) to popularna technika mająca na celu podział warstwy biznesowej na dwie części: *Serwis* i *Repozytorium*. *Serwis* jest elementem odpowiedzialnym za logikę aplikacji oraz komunikację między kontrolerem a repozytorium. *Repozytorium* natomiast jest komponentem, ktorego zadanie polega na komunikacji z bazą danych: zapisywanie, pobieranie, edytowanie i usuwanie danych (model *CRUD*). Takie rozszczepienie poszczególnych elementów pozwala na łatwe testowanie kodu, programowanie zorientowane na testy (*ang. Test Driven Development*), szybką modernizację istniejącej logiki i rozbudowę aplikacji.

W projekcie *Wzorzec Repozytorium* został zaimplementowany poprzez biblioteki **PI.Service (Serwis)** i **PI.Data (Repozytorium)**. Dane z kontrolerów przekazywane są do *Serwisu* pod postacią *ViewModles* lub typów prymitywnych. **PI.Service** odpowiada za odpowiednią logikę przetwarzania danych, konwersji *ViewModels* na *Models* (przy wykorzystaniu narzędzia *AutoMapper* [1]) i przekazaniu ich do *Repozytorium*.

Funkcją biblioteki **PI.Data** jest odbieranie danych z *Serwisu* i dokonanie odpowiednich operacji bazodanowych przy użyciu *EntityFramework* [2]

[1] AutoMapper - Sekcja 2.10.2
[2] EntityFramework - Sekcja 2.10.1

Schemat działania logiki biznesowej w oparciu o *Wzorzec Repozytorium* został przedstawiony na Rysunku 3.2 .




Rys. 3.2: Schemat przepływu danych przy wykorzystaniu Wzorca Repozytorium

### 3.1.3. Odwrócenie sterowania

Odwrócenie sterowania (*ang. Inversion of Control*) to paradygmat odpowiedzialny za przeniesienie na zewnątrz obiektu odpowiedzialnego za kontrolę niektórych czynności. Termin ten jest często utożsamiany z *Wstrzykiwaniem zależności (ang. Dependency Injection)*, jednak jest to tylko jedna z realizacji *Odwrócenia sterowania*.

Wstrzykiwanie zależności zosatało zrealizowane w projekcie za pomocą biblioteki *Unity* [3] na dwóch poziomach:

- w warstwie Modelu - wszystkie serwisy i repozytoria oraz kontekst bazodanowy wstrzykiwane są jako parametry w konstruktorze,

[3] Unity - Sekcja 2.10.3

- w warstwie Kontrolorów - wstrzykiwany jest dostawca serwisów.

Takie rozdzielenie wstrzykiwania zależności pozwoliło na całkowite odseparowanie logiki bazodanowej i elementów za nią odpowiedzialnych od warstwy aplikacji.

### 3.1.4. Bezpieczeństwo

#### Autoryzacja

Do autoryzacji użytkowników w projekcie wykorzystany został popularny z technologii ASP.NET system *Forms authentication*. *Forms authentication* wykorzystuje znacznik uwierzytelniający, który zostaje utworzony w momencie gdy użytkownik zaloguje się na stronie. Znacznik *Forms authentication* zwykle przechowywany jest wewnątrz szyfrowanego ciasteczka.

W momencie gdy użytkownik będzie chciał uzyskać dostęp do strony wymagającej uwierzytelnienia, a nie przechodził wcześniej procesu autoryzacji, zostanie przekierowany na zdefiniowaną w pliku konfiguracyjnym stronę logowania, na której będzie mógł wprowadzić nazwę użytkownika i hasło. Te dane są następnie przekazane do serwera, który sprawdza czy takowy użytkownik istnieje w bazie danych. Po pomyślnej weryfikacji danych, użytkownik zostaje uwierzytelniony, stworzony zostaje znacznik i następuje przekierowanie na stronę główną aplikacji.

#### Przechowywanie hasła

Przechowywane w systemie hasła, szyfrowane są poprzez wygenerowanie funkcji skrótu *SHA-2* o wielkości 256 sklejonego z *solą* hasła. *Sól* stanowi wygenerowany w momencie rejestracji *GUID - identyfikator globalnie unikatowy*. W aplikacji został wykorzystany algorytm *SHA-2*, ponieważ jest zancznie silniejszy od swoich poprzedników z rodziny *SHA-1*, w którym zidentyfikowano luki w bezpieczeństwie. Ogólny algorytm szyfrowania wygląda następująco:

Listing 3.1: Szyfrowanie hasła

```
SHA256(concat(password, salt))
```

Dzięki zastosowaniu *soli* zmniejszone zostaje prawdopodobieństwo złamania hasła popularnymi atakami: brutal force, atakiem słownikowym czy wykorzystującym tablice tęczowe.

### 3.1.5. Wielojęzykowość

Dzięki zastosowaniu mechanizmu *Zasobów (ang. Resources)* aplikacja internetowa jest dostosowana do szybkiego tłumaczenia na inne języki. Domyśnym językiem aplikacji jest angielski, a w obecnej wersji obsługuje również polski. By dodać inne tłumaczenie, należy stworzyć nowy plik *resx* o nazwie:

Listing 3.2: Tłumaczenie aplikacji

```
Main.xx.resx
```

gdzie *xx* to dwuliterowy skrót języka przeglądarki, w którym należy umieścić odopwiednie wartość dla podaych kluczy.

### 3.1.6. Azure

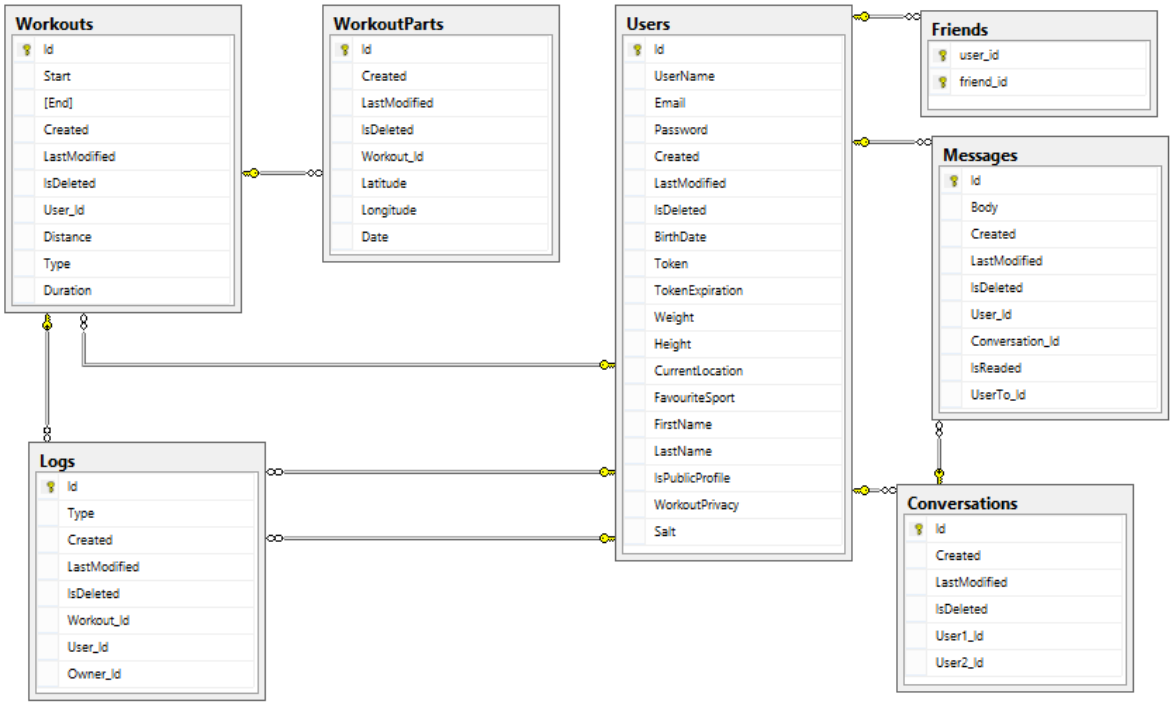
Aplikacja została przystosowana do natychmiastowej możliwości wdrożenia na platformę Windows Azure SDK. W tym celu należy dokonać publikacji projektu o nazwie **PI.Azure** oraz zaimportować go jak serwis lub stronę w portalu http://windos.azure.com.

## 3.2. Baza danych

Baza danych, do której dostęp ma aplikacja internetowa działa na serwerze *Microsoft SQL Server 2012* [4]. Do zarządzania danymi wykorzystany został wykorzystana bibliotek *EntityFramework* [5]. W bazie danych przechowywane są informacje o profilu użytkownika, treningach, wiadomościach, logach, ustawieniach konta oraz zarejestrowane przez aplikację mobilną współrzędne GPS.

W tworzeniu i zarządzaniu bazą danych wykorzystane zostało podejście *Code First*, polegające na modelowaniu bazy danych pod postacią klas języka *C#*, a następnie wygenerowaniu gotowej bazy danych przy pomocy EntityFramework. By zabezpieczyć dane składowane w bazie danych przed usunięciem ich w wyniku zmiany modelu w kodzie projektu, użyty został mechanizm migracji. Schemat bazy danych przedstawiony został na Rysunku 3.3 .

[4] MS SQL 2012 - Sekcja 2.5.1
[5] EntityFramework - Sekcja 2.10.1




Rys. 3.3: Schemat bazy danych

## 3.3. Warstwa prezentacji

Aby aplikacja internetowa była przyjazna i łatwa w użyciu, należy zadbać o przejrzysty, łatwy w obsłudze i estetyczny interfejs. W tym celu wykorzystana została popularna biblioteka stylów CSS Twitter-Bootstrap [6]. Dzięki dużej ilośći gotowych komponentów, można w łatwy i szybki sposób projektować wygląd strony internetowej, spełniający oczekiwania współczesnego użytkownika. Twitter-Bootstrap to nie tylko biblioteka styli CSS, ale również zbiór elementów JavaScript takich jak okna modalne czy zakładki.

W celu poprawienia funkcjonalności aplikacji i interakcji z użytkownikiem wykorzystane zotały dodatkowe moduły takie jak:

- jQuery Data Table [7],
- pNotify [8],
- jQuery DatePicker [9],
- jQuery DateTimePicker [10],

[6] http://twitter.github.com/bootstrap/
[7] http://datatables.net/
[8] http://pinesframework.org/pnotify/
[9] http://jqueryui.com/
[10] http://trentrichardson.com/examples/timepicker/

- FullCalendar [11],
- Google Maps JavaScript API v3 [12].

## 3.4. Funkcjonalności

### 3.4.1. Treningi

Moduł do obsługi treningów jest najważniejszym elementem projektu. Pozwala on na śledzenie, dodawanie i archiwizowanie danych dotyczących aktywności fizycznych. Zapisane w systemie treningi wyświetlane są w kalendarzu (Rysunek 3.4), przez co można w uporządkowany sposób zarządzać treningami, planować następne sesje, archiwizować dane i wyszukiwać wcześniejsze rekordy. Po kliknięciu na wybrany przez nas trening, wyświetlone zostaje modalne okno zawierające wszystkie informacje dotyczące aktywności: typ, data początku i końca, czas, dystans i jeżeli w bazie danych są przechowane współrzędne GPS, to również mapa z zaznaczoną trasą (Rysunek 3.5).

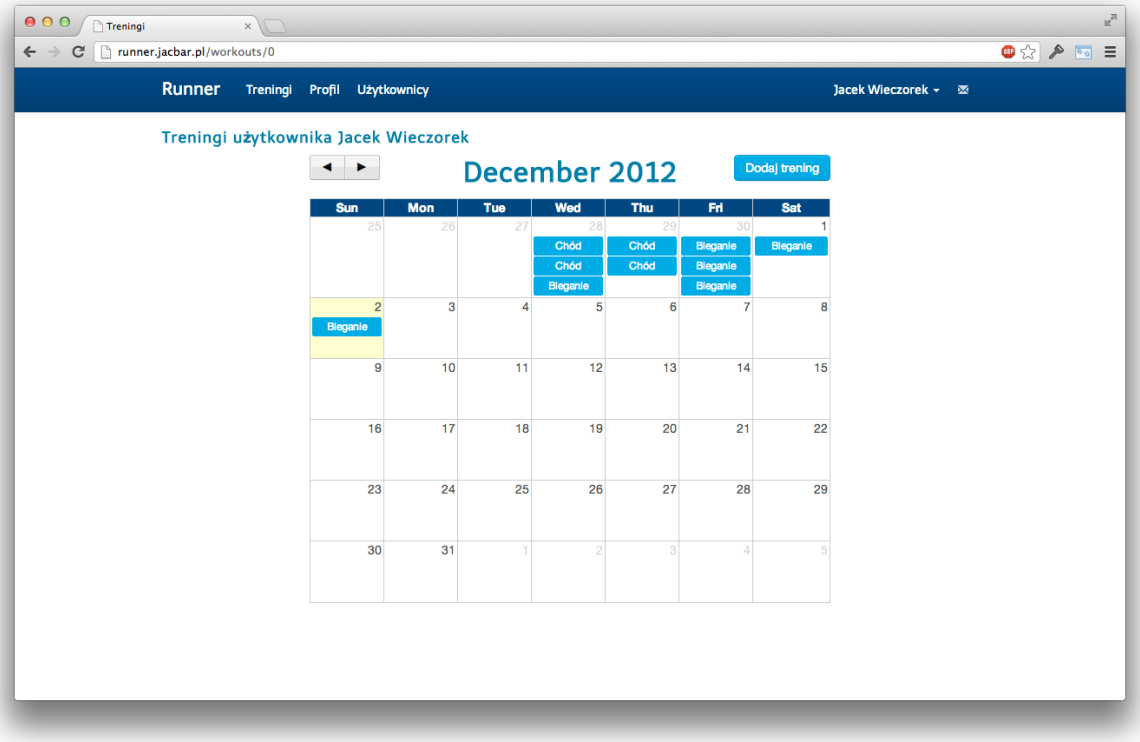
Dodawanie treningu może odbywać się na dwa sposoby: poprzez synchronizację z danymi przechowywanymi przez aplikację mobilną lub dodając rekord bezpośrednio przez interfejs aplikacji internetowej (Rysunek 3.7).

## 3.5. Funkcje portalu społecznościowego

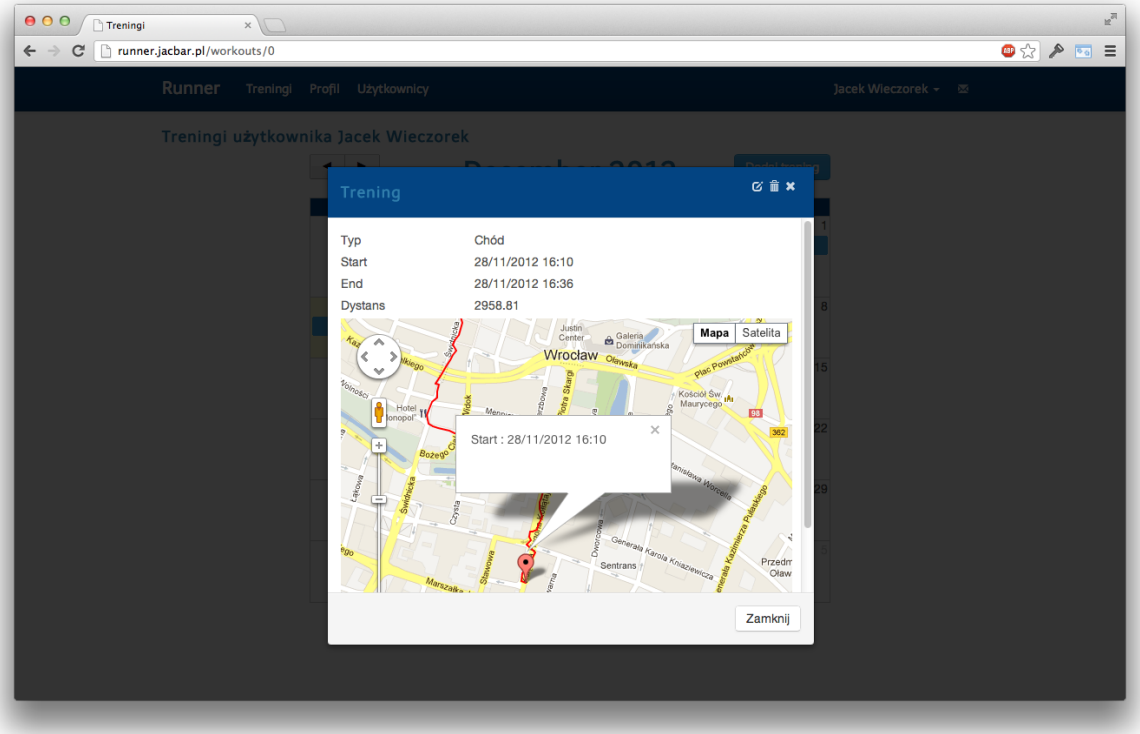
By zapewnić powodzenie aplikacji w dzisiejszych czasach należy zadbać również o dodatkowe funkcjonalności aplikacji, pozwalające na interakcję z użytkowników między sobą. Możliwość podążania za użytkownikami, przeglądanie ich aktywności, osiągnięć, możliwość komunikacji poprzez wysyłanie wiadomości sprawia, iż aplikacja staje się małym portalem społecznościowym.

Aby zadowolić użytkowników, chcących wykorzystywać serwis tylko i wyłącznie jako internetowy dziennik treningów, zaimplementowane zostały różne poziomy udostępniania treści innym osobom korzystającym z portalu. Podstawowe wiadomości jak imię i nazwisko oraz nazwa użytkownika są widoczne dla wszystkich, jednak resztę informacji w tym adres email można ukryć, wyłączając w widoku edycji profilu flagę "Profil publiczny". Istnieje również możliwość udostępniania swoich treningów innym użytkownikom na trzech poziomach: dostępne dla wszystkich, dostępne dla użytkowników podążających, niewidoczne dla nikogo.

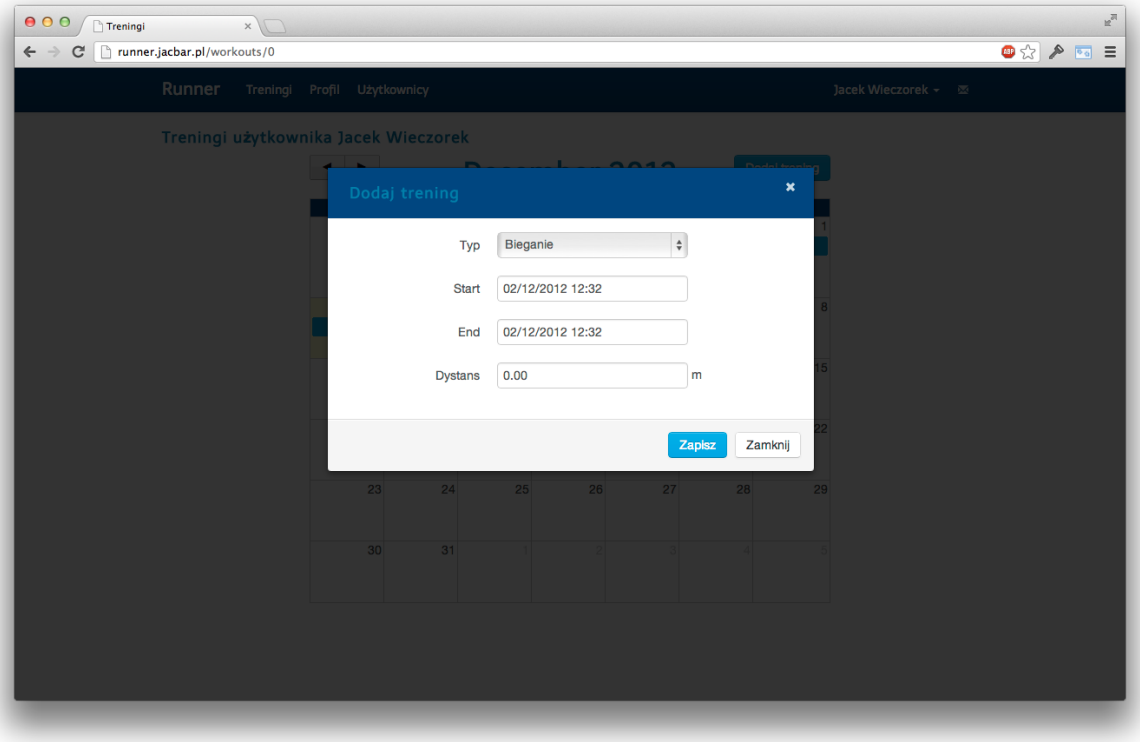
[11] http://arshaw.com/fullcalendar/
[12] https://developers.google.com/maps/documentation/javascript/



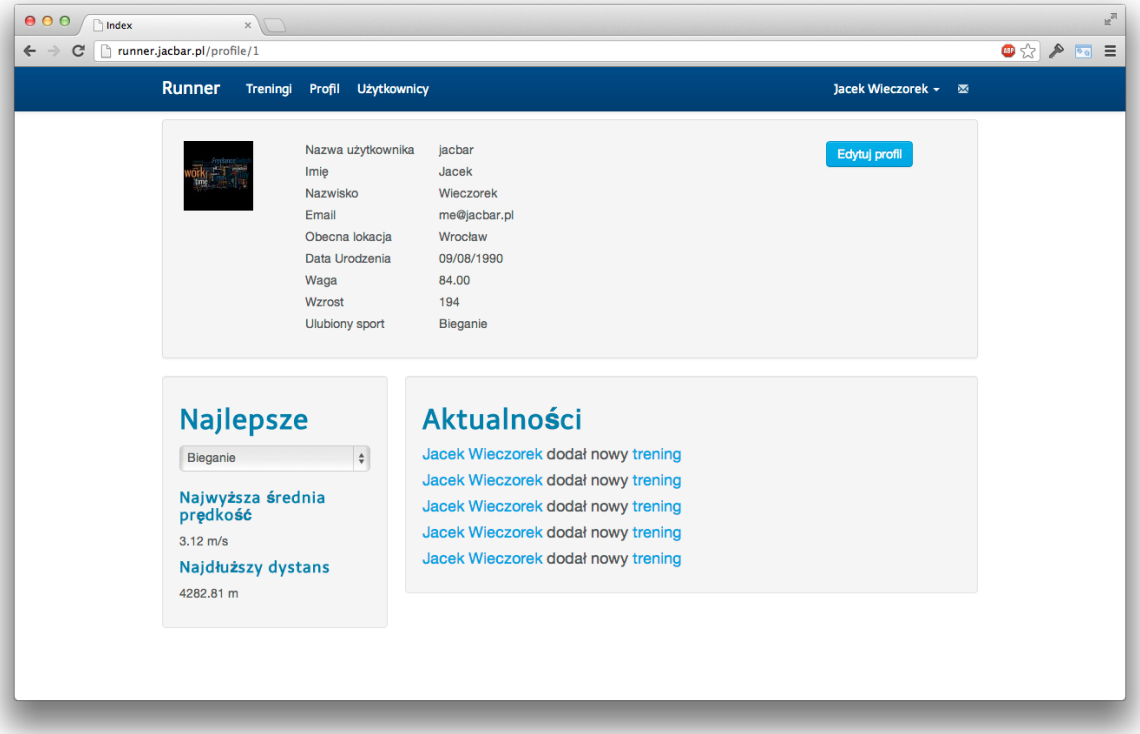

Rys. 3.4: Widok strony agregującej dodane rekordy




Rys. 3.5: Widok treningu z zaznaczoną trasą




Rys. 3.6: Dodawanie treningu przez aplikację internetową




Rys. 3.7: Profil użytkownika

# Rozdział 4

# Aplikacja mobilna

Aplikacja mobilna jest integralną częścią całego systemu do wspierania treningów areobowych. Jej zadaniem jest pobieranie informacji o przebytej trasie za pomocą odbiornika GPS, obliczanie dystansu i czasu trwania treningu, przetrzymywanie informacji w bazie danych oraz synchronizacja rekordów z apikacją internetową.

## 4.1. Architektura systemu

Aplikacja została napisana w języku Java i działa na mobilnej platformie Android w wersji 4.0.3. Najważniejszą częścia programów napisanych na system Android jest *Aktywność (ang. Activity)*, charakteryzująca się określonym cyklem życia. Sposób w jaki zarządza się cyklem życia aplikacji, uruchamia odpowiednie zdarzenia w określonych momentach i zachowanie aktywności ma fundamentalny wpływ na działanie całej aplikacji.

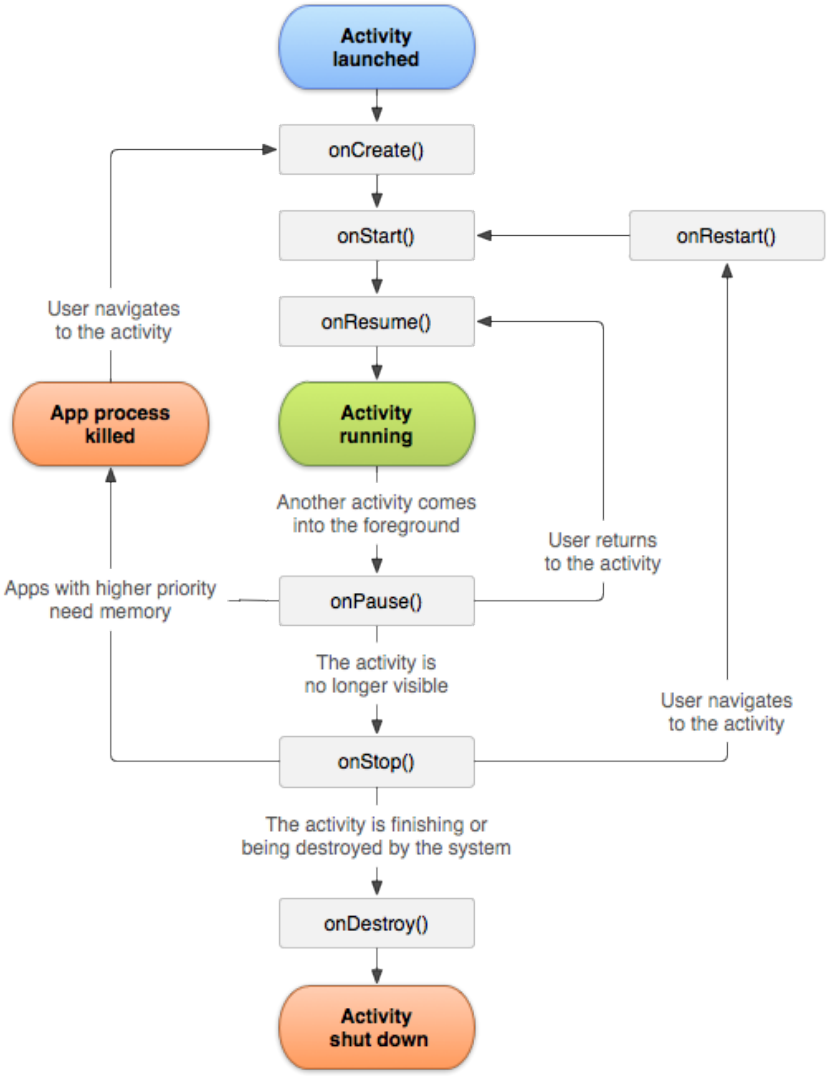
### 4.1.1. Aktywności

Aktywności wyróżniają się czterema stanami:

- aktywna (ang. active) lub chodząca (ang. running) - gdy aktywność znajduje się w warstwie szczytowej aplikacji,
- wstrzymana (ang. paused) - gdy aktywność jest widoczna, ale nie znajduje się na pierwszym planie. W tym stanie może zostać zakończona przez system, w przypadku braku pamięci,
- zatrzymana (ang. stopped) - gdy aktywność zostanie zasłonięta przez inna aktywność. W tym stanie system przechowuje pełną informację na temat aktywności, lecz nie jest widoczna dla użytkownika,

- gdy aplikacja jest w stanie wstrzymania lub zatrzymania, może zostać usunięta z pamięci systemu. W momencie ponownego uruchomienia, aktywność będzie musiała być stworzona od początku.

Pełny cykl życia aktywności został przedstawiony na Rysunku 4.1



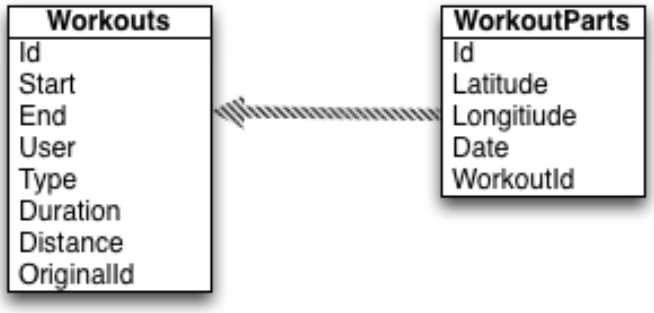

Rys. 4.1: Pełny cykl życia aktywności

### 4.1.2. Moduł GPS

Obsługa modułu GPS polega na zaimplementowaniu interfejsów odpowiedzialnych za zczytywanie informacji (*LocationListener*) na temat położenia i badanie stanu odbiornika GPS (*GpsStatusListener*). Oba interfejsy są uruchamiane w zdarzeniu *onCreate()* głównej aktywności.

### 4.1.3. Baza danych

Informacje na temat treningów przechowywane są w bazie danych SQLite [1]. Model bazy danych (Rysunek 4.2) stanowią dwie tabele: *Workout* i *WorkoutDetails*. Pierwsza z nich służy do przechowywania podstawowych informacji o treningu: początek, koniec, czas trwania, dystans. W drugiej tabeli natomiast przechowywane są współrzędne geograficzne zczytane z GPS.




Rys. 4.2: Schemat bazy danych w aplikacji mobilnej

### 4.1.4. Autoryzacja

Po wprowadzeniu nazwy użytkownika i hasła w formularzu na aktywności Login, wysłane zostaje rządanie autoryzacji do REST API. Po pozytywnej weryfikacji danych zwrócona zostaje odpowiedź o statusie HTTP 200 wraz z tokenem, służącym do dalszej autoryzacji aplikacji. Zweryfikowane dane użytkownika (login i hasło) przechowywane są w *SharedPreferences*. Szczegółowy opis autoryzacji znajduje się w sekcji 5.2.1.

### 4.1.5. Wielojęzykowość

Podobnie jak strona internetowa, aplikacja mobilna została dostosowana do wielojęzykowości, dzięki popularnemu w środowisku android systemowi zasobów.

### 4.1.6. Wyświetlanie mapy

W celu wyświetlenia przebytej trasy w aplikacji mobilnej wykorzystane zostało *Google Maps Android API*. Pozwala ono na pobranie z serwisu Google mapy o wskazanej pozycji i przybliżeniu, a także na rysowaniu lini pomiędzy punktami na jednej z jej warstw.

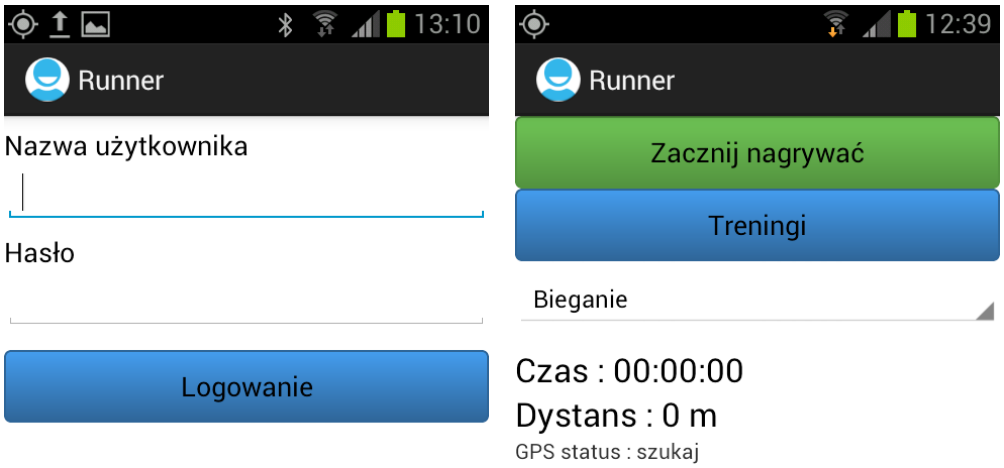
Aby wyznaczyć środek wyświetlanego obszaru, trzeba znaleźć cztery wartości: maksymalną i minimalną długość i szerokość. Punkt środka mapy wyznacza się z następującego wzoru:

[1] SQLLite - Sekcja 2.5.2

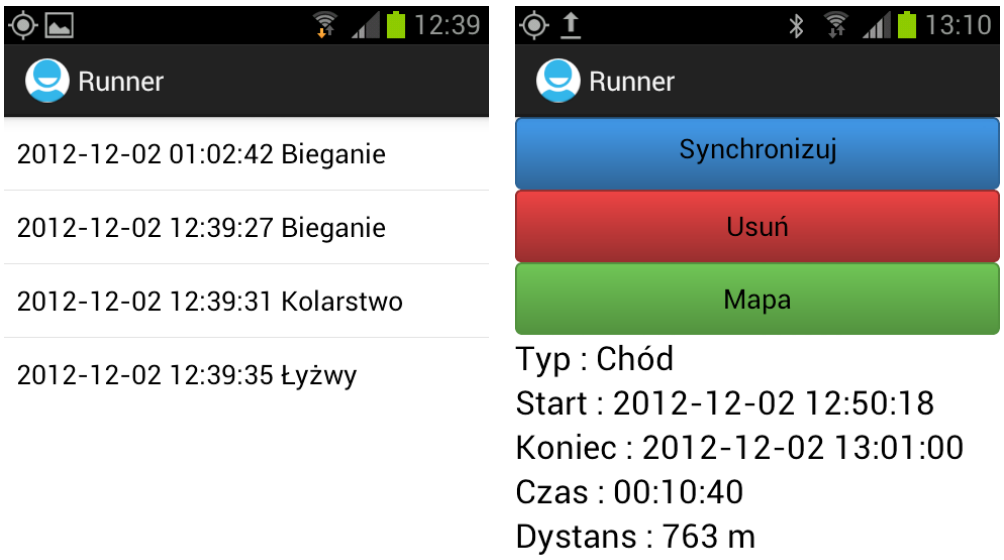
$$CenterLatitude = \frac{MaxLatitude + MinLatitude}{2} \tag{4.1}$$

$$CenterLongitude = \frac{MaxLongitiude + MinLongitiude}{2} \tag{4.2}$$

Następnym etapem jest pobranie z bazy danych współrzędnych zarejestrowanych punktów i połączenie ich za pomocą linii.




Rys. 4.3: Po lewej ekran logowania, po prawej ekran główny apliakcji




Rys. 4.4: Po lewej lista treningów, po prawej szczegółowy widok treningu


Rys. 4.5: Widok zaznaczonej na mapie trasy

# Rozdział 5

# REST Api

## 5.1. Architektura

*REST*, czyli Represential State Transfer jest wzorcem opartym na doświadczeniach z tworzenia protokołu *HTTP 1.0*. W dzisiejszych czasach *REST* stał się dominującym modelem do projektowania usług sieci WEB. Opiera się na czterech głównych metodach znanych z protokołu *HTTP*:

- GET - pobieranie daych,
- POST - tworzenie danych,
- PUT - modyfikowanie danych,
- DELETE - usuwanie danych.

Systemy w tej architekturze składają się z przynajmniej dwóch elementów: klienta i serwera. Aplikacje klienckie wysyłają rządania do serwera, który odpowiada za przetworzenie ich i zwrócenie odpowiedzi. *World Wide Web* jest największym znanym przykładem systemu zgodnego z architektuą *REST*.

*RESTowe* Web Services, zwane także API, są serwisami zaimplementowanymi przy użyciu protokołu *HTTP* i głównych założeń *REST*. Standardowe serwisy składają się z:

- bazowego *URI* (Uniform Resource Identifier) identyfikującego serwis np. http://runner.jacbar.pl/Api.svc,
- internetowego nośnika danych obsługiwanego przez serwis, najczęściej *XML* lub *JSON*,
- zbioru operacji bazujących na podstawowych metodach protokołu *HTTP*.

W projekcie do zaimplementowania *RESTowego API* wykorzystana została popularna technologia wspierana przez Microsoft - *Windows Comunication Foundation* [1]. Jako nośnik danych użyty został czytelny dla człowieka format zapisu danych *JSON*. Api zostało stworzone w celu umożliwienia aplikacji mobilnej komunikację z bazą danych, zapisywaniem do niej informacji i autoryzacją użytkowników.

Podobnie jak w przypadku apliakcji internetowej, również w serwisie zastosowany został Wzorzec Repozytorium.

## 5.2. Komunikacja API <-> Aplikacja mobilna

### 5.2.1. Autoryzacja

Autoryzacja w aplikacji przebiega w dwóch etapach: zalogowanie się i autoryzacja rządań z wykorzystaniem tokenu.

#### Proces logowania

W celu poprawnej identyfikacji użytkownika starającego się uzyskać dostęp do aplikacji mobilnej, wysyłane jest rządanie identyfikacji do API. W tym etapie wykorzystana została podstawowa metoda autoryzacji - *Basic Authentication*. Do wysyłanego requestu wstawiany jest nagłówek *RunnerAuthorization*, którego wartość wyznaczana jest w następujący sposób:

Listing 5.1: Parametr nagłówka rządania autoryzacji

```
RunnerAuthorization : Base B64(username:password)
```

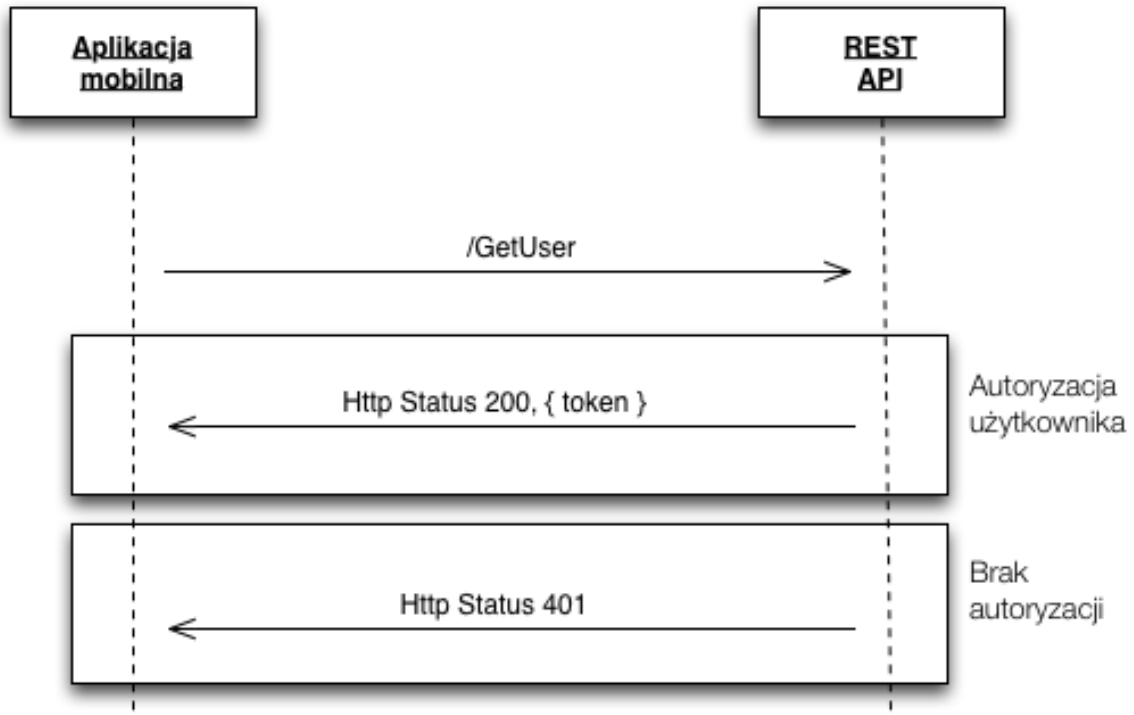
Gdy proces dekodowania danych i autoryzacji użytkownika przebiegnie pomyślnie (Rysunek 5.1), zwracana jest odpowiedź o statusie HTTP 200 zawierająca token, ktorego okres życia wynosi 30 minut. Od tego czasu aplikacja w celach autoryzacji posługuje się tokenem. Każdy kolejne zautoryzowane rządanie z apliakcji mobilnej do web serwisu ustawia ważność tokenu na 30 minut. W przypadku, gdy okres ten zostanie przekroczony, API zwraca odpowiedź o statusie 401. W tym momencie aplikacja mobilna pobiera zapisane w *SharedPreferences* login i hasło, a następnie wysyła podstawowe rządanie autoryzacji. W przypadku braku autoryzacji ze strony serwera lub braku danych autentykujących w *SharedPreferences*, użytkownik przenoszony jest na aktywność logowania. Szczegółowy opis parametrów rządania autoryzacji przedstawiony został w tabeli 5.1 .

[1] Windows Comunication Foundation - Sekcja 2.4

Tab. 5.1: Rządanie autoryzacji

| | |
|---|---|
| Metoda | GET |
| URI | http://runner.jacbar.pl/Api.svc/GetUser |
| Autoryzacja | Basic Authentication |
| HTTP Status | 200 - ok<br>401 - brak autoryzacji<br>400 - nieprawidłowe rządanie |
| Przykładowa odpowiedź | {<br>”username” : ”jacek”<br>”token” : ”030B4A82-1B7C-11CF-9D53-00AA003C9CB6”<br>} |

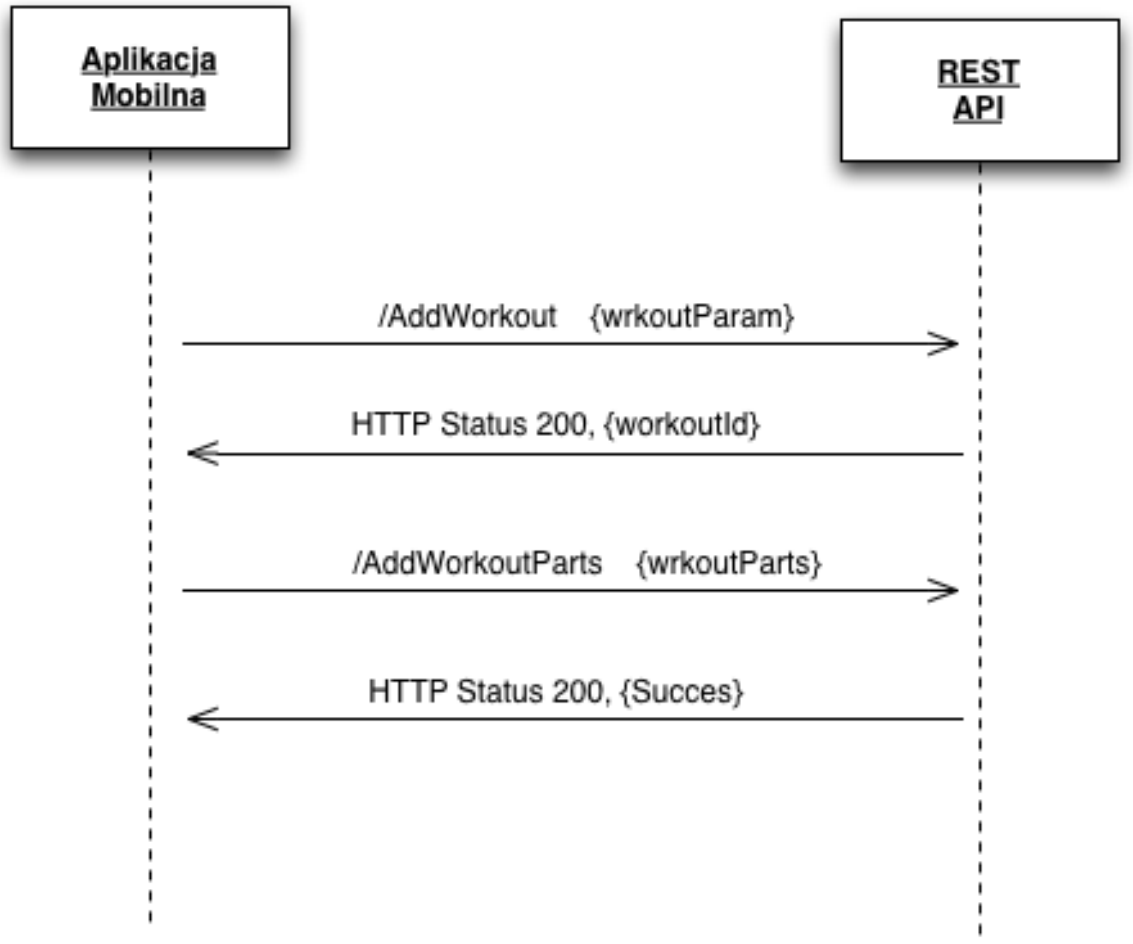
By zapewnić bezpieczeństwo tego rozwiązania, należy zadbać o szyfrowanie połączenia za pomocą standardu *SSL*, np. poprzez wykorzystanie szyfrowanego protokołu *HTTPS*.




Rys. 5.1: Schemat autoryzacji użytkownika

### 5.2.2. Synchronizacja treningów

W celu umożliwienia użytkownikowi zarządzania treningami, nagranymi za pomocą aplikacji mobilnej, w serwisie internetowym web serwis posiada metody pozwalające na przesyłanie danych z urządzenia na serwer. Proces synchronizacji polega na wysłaniu dwóch rządań. Pierwszego w celu stworzenia nowego treningu w bazie na serwerze i wygenerowaniu jego unikalnego identyfikatora. Drugiego przesyłającego pozycje zczytane z GPS, jeżeli takowe zostały zapisane w pamięci telefonu. Celem rozdzielenia tych dwóch procesów jest zapewnienie w przyszłości możliwości wysyłania pozycji GPS w paczkach. Pozwoli to na zmniejszenie ryzyka utraty danych oraz ułatwi proces retransmisji danych, które nie zostały zapisane w bazie danych na serwerze. Szczegółowe parametry dotyczące wysyłanych rządań przedstawione zsotały w tabelach 5.2 i 5.3 .




Rys. 5.2: Schemat synchronizacji treningów

Tab. 5.2: Tworzenie treningu na serwerze

| | |
|---|---|
| Metoda | POST |
| URI | http://runner.jacbar.pl/Api.svc/AddWorkout |
| Autoryzacja | token |
| HTTP Status | 200 - ok<br>401 - brak autoryzacji<br>400 - nieprawidłowe rządanie |
| Przykładowe dane | {<br>”Distance” : ”2000”,<br>”Duration": ”300”,<br>”Start” : ”2012-12-03 23:45:00”,<br>”End” : ”2012-03-12 23:59:00”,<br>”Type” : ”1”<br>} |
| Przykładowa odpowiedź | {<br>”workoutId” : ”5”<br>} |

Tab. 5.3: Synchronizacja zarejestrowanych współrzędnych geograficznych

| | |
|---|---|
| Metoda | POST |
| URI | http://runner.jacbar.pl/Api.svc/AddWorkoutParts |
| Autoryzacja | token |
| HTTP Status | 200 - ok<br>401 - brak autoryzacji<br>400 - nieprawidłowe rządanie |
| Przykładowe dane | {<br>”WorkoutId” : ”5”,<br>”WorkoutParts” : [<br>{<br>”Latitiude” : ”51.09309479832”,<br>”Longitude” : ”51.09309479832”,<br>”Date” : ”2012-03-12 23:57:00”<br>},<br>{<br>”Latitiude” : ”51.09309479832”,<br>”Longitude” : ”51.09309479832”,<br>”Date” : ”2012-03-12 23:57:10”<br>}<br>]<br>} |
| Przykładowa odpowiedź | {<br>”Success” : ”true”<br>} |

# Rozdział 6

# Optymalizacja wydajności aplikacji

W celu poprawienia wydajności aplikacji oraz sprawdzenia zachowania w przypadku obciążenia dużą liczbą jednoczesnych zapytań pobierających rekordy z bazy danych, przeprowadzone zostały testy obciążeniowe. Jako środowisko testowe, wykorzystany został program *Gatling Tool* [1]. Gatling Tool jest wolnym oprogramowaniem cechującym się:

- wysoką wydajnością,
- prostotą założeń,
- wsparciem dla protokołu *HTTP*,
- możliwością nagrywania scenariuszy,
- złożoną analizą rezultatów.

Gatling Tool jest programem napisanym w języku *Scala* [2] i działającym na wirtualnej maszynie javy. Dzięki temu, teoretycznie, testy można przeprowadzić w każdym systemie wspierającym technologie *JVM*. Komputer testowy miał następujące parametry:

- procesor: Intel Pentium i5 - 2,3 GHz,
- pamięć RAM: 8GB DDR3,
- sytem operacyjny: OSX 10.8 Mountain Lion.

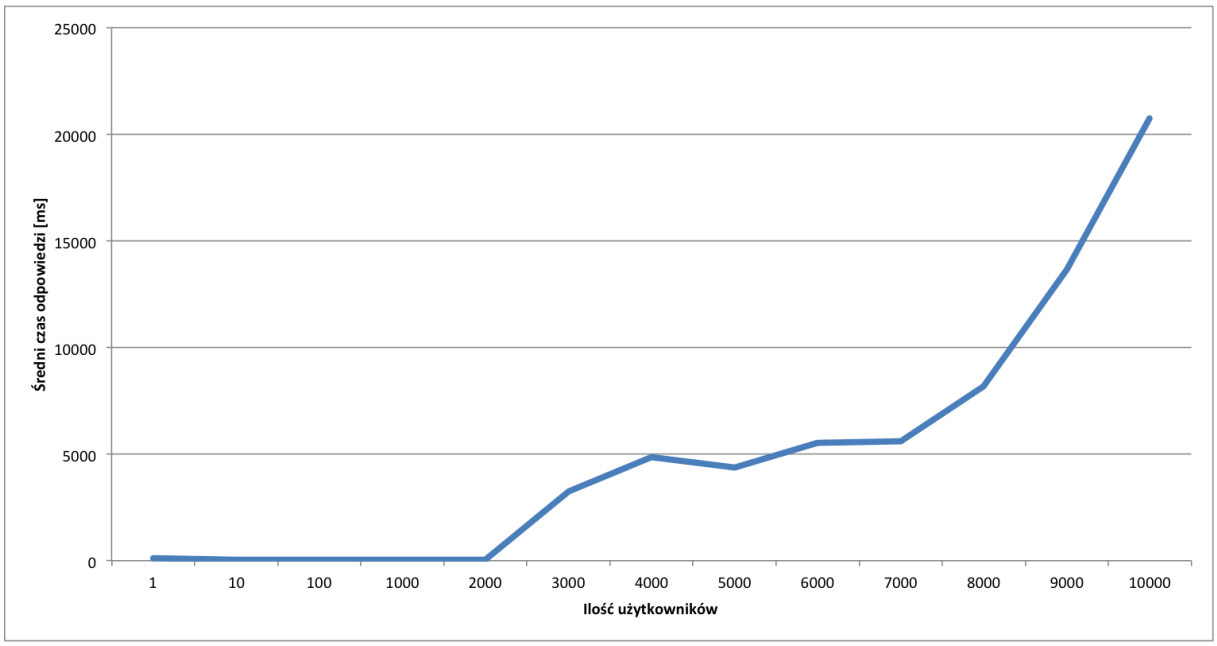
Aplikacja internetowa została wdrożona na serwerze o następujących parametrach:

- procesor: Intel Pentium Dual Core 1,83 GHz,
- pamięć RAM: 3 GB DDR2,

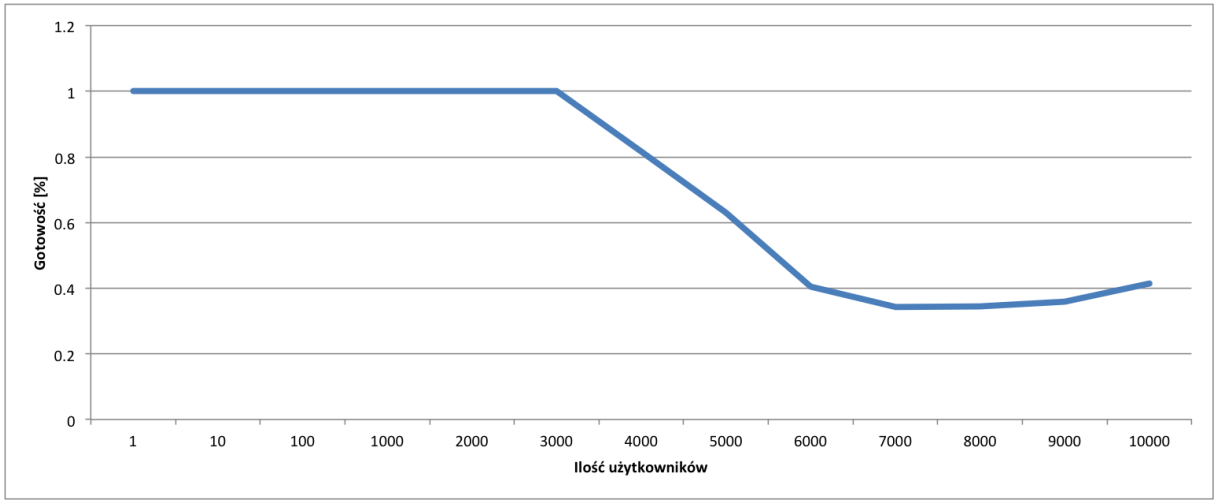
[1] Gatling Tool - http://gatling-tool.org/
[2] Scala - http://scala.org

- system operacyjny: Windows 7,
- oprogramowanie: IIS 7, Microsoft SQL Server 2012.

Scenariusz testów obejmował załadowanie strony głównej aplikacji, a następnie wyświetlenie profilu o id=1 i id=2. Testy wykonane zostały dla zmiennej liczby użytkowników od 1 do 10000 (1, 10, 100, 1000, 2000, ..., 10000). Wyniki symulacji przedstawione zostały na Rysunkach 6.1 i 6.2

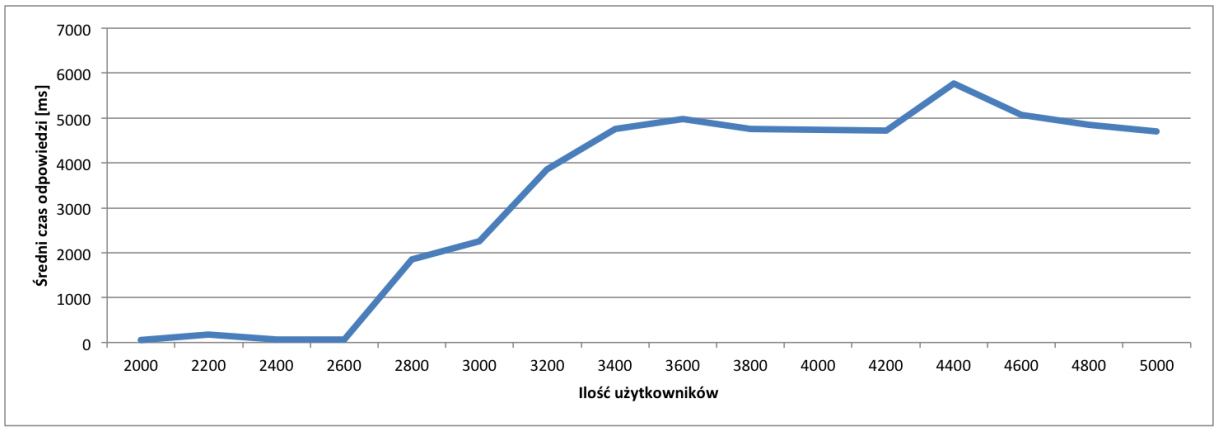



Rys. 6.1: Przebieg średniego czasu odpowiedzi dla całego testu

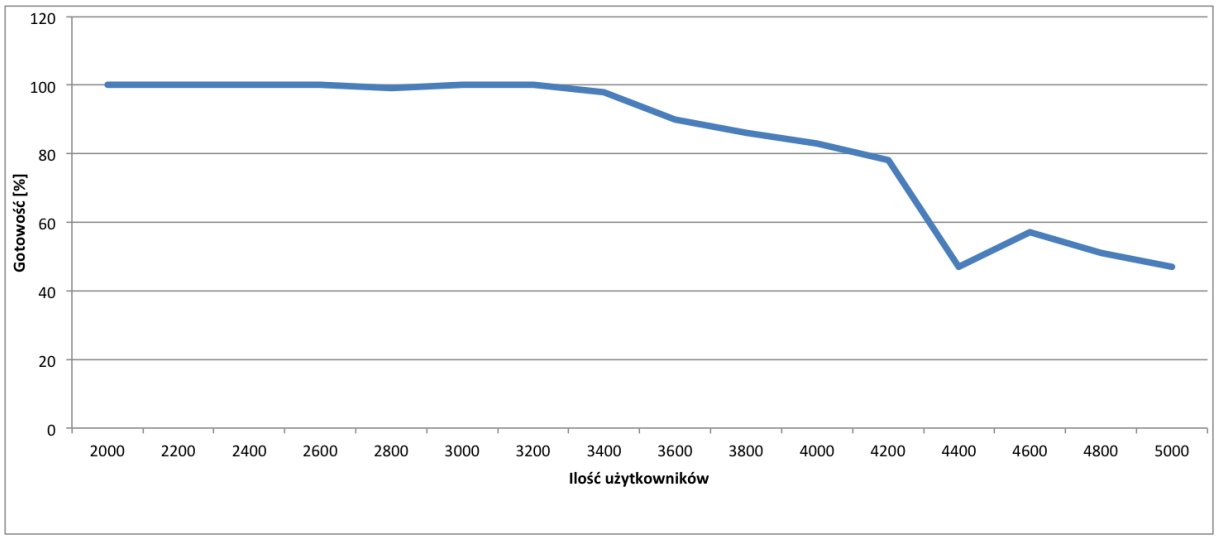



Rys. 6.2: Przebieg gotowości systemu dla całego testu

Z danych odczytanych na wykresach (Rysunek 6.1 i 6.2) można zauważyć, że istotny z punktu poprawy wydajności systemu jest zakres jednoczesnych użytkowników systemu od 2000 - 5000. Powyżej 7000 użytkowników, komputer testowy nie nadążał obsługiwać wszystkich przychodzących rządań, powodując błędy programu testowego. Na Rysunkach 6.3 i 6.4 przedstawiono wyniki dokładniejszego próbkowania zakresu użytkowników 2000 - 5000.




Rys. 6.3: Przebieg średniego czasu odpowiedzi dla zakresu użytkowników od 2000 do 5000

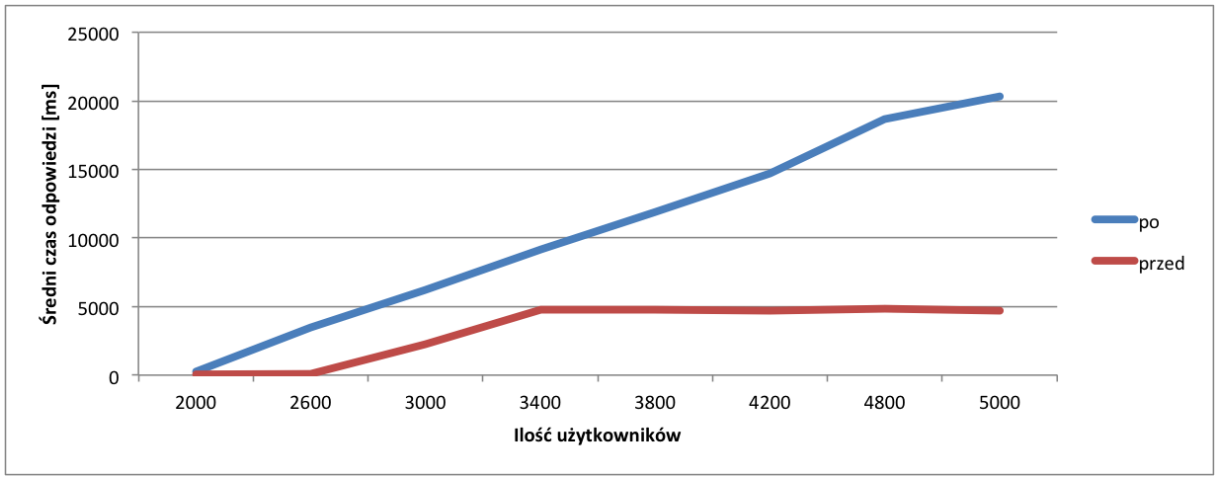



Rys. 6.4: Przebieg gotowości systemu dla zakresu użytkowników od 2000 do 5000

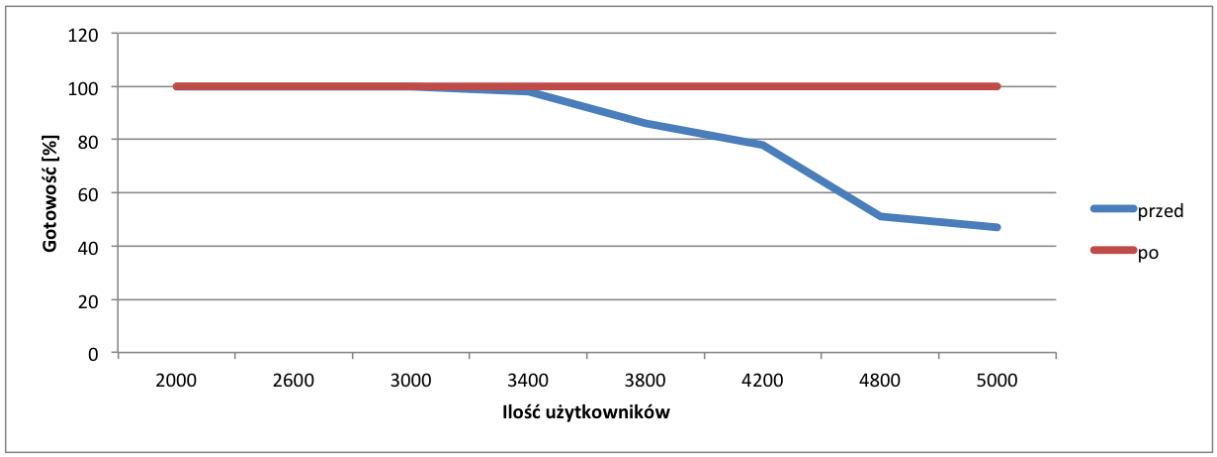
Powyższe testy zostały wykonane przy wykorzystaniu bazy danych, która została zindeksowana przez *EntityFramework* [3] przy kreowaniu jej. Poprawa wydajności aplikacji została osiągnięta poprzez modyfikację ustawień serwera IIS - zwiększenie czasu odpowiedzi na rządania i przetwarzanej ich liczby. W tym celu wykorzystany został program *IIS Tuner* [4], służący

[3] EntityFramework - 2.10.1
[4] IIS Tuner - http://iistuner.codeplex.com/

do optymalizacji puli aplikacji. Wyniki symulacji po rekonfiguracji ustawień serwera przedstawione zostały na Rysunkach 6.5 i 6.6. Średni czas odpowiedzi serwera znacząco wzrósł, ale za to gotowość systemu utrzymywała się cały czas na stu procentowym poziomie.




Rys. 6.5: Przebieg średniego czasu odpowiedzi dla zakresu użytkowników od 2000 do 5000 po rekonfiguracji serwera




Rys. 6.6: Przebieg gotowości systemu dla zakresu użytkowników od 2000 do 5000 po rekonfiguracji serwera

# Rozdział 7

# Podsumowanie

## 7.1. Realizacja założeń

Przedstawiony projekt w pełni realizuje założenia przedstawione w Sekcji 1.2. Jest całkowicie funkcjonalnym i działającym systemem, pozwalającym na kontrolowanie swoich treningów oraz interakcje z innymi użytkownikami portalu.

## 7.2. Napotkane problemy

Głównym problemem napotkanym podczas implementacji i testowania systemu była konfiguracja serwera *IIS*. Przy użyciu domyślnych parametrów serwer nie zezwalał na sposoby autoryzacji zastosowane w projekcie.

## 7.3. Kierunki rozwoju

By zapewnić sukces marketingowy projektu, należałoby zaimplementować aplikacje mobilne na inne systemy operacyjne niż tylko *Andorid*. W dzisiejszych czasach jednymi z najpopularniejszych systemów operacyjnych są: iOS, Android, BlackBerry OS, a w ostatnich czasach również Windows Phone 7.5 i 8. Nie można także zapomnieć o użytkownikach posiadających starsze telefony działające na systemie Symbian OS, Bada OS.

W celu zapewnienia bezpieczeństwa przesyłanych danych pomiędzy aplikacją mobilną, a REST API, system powinien zostać uzupełniony o możliwość szyfrowania danych poprzez wykorzystanie certyfiaktów *SSL*.

Ważnym elementem jest również niezawodność działania aplikacji internetowej. By zapewnić skalowalność serwera i łatwą konfigurowalność parametrów instancji na jakich działa system, powinien zostać wdrożony na platformę Windows Azure, do czego został dostosowany.

Z uwagi na niską jakość montowanych w urządzenaich mobilnych odbiorników GPS, a w konsekwencji niską dokładność odczytywanych pozycji, należałoby w celu wyświetlania na mapie przebytej trasy zaimplementować filtr Kalmana. Pozwoliłoby to na pokazanie wygładzonej trasy z mniejszą ilością niedokładności spowodowanych dużą rozdzielczością odbiornika.

# Literatura


[1] Android. http://developer.android.com/guide/components/index.html, listopad 2012.

[2] Android. http://developer.android.com/reference/packages.html, listopad 2012.

[3] Pablo Cibaro, Kurt Clayes, Fabio Cozzolino, and Johann Grabner. *Professional WCF 4: Windows Communication Foundation in .NET 4.0*. Wiley Publishing, Inc., 2010.

[4] Adam Freeman and Steven Sanderson. *Pro ASP.NET MVC 3 Framework*. Wiley Publishing, Inc., 2011.

[5] Zigurd Mednieks, Laird Dornin, G. Blake Meike, and Masumi Nakamura. *Programming Android*. O'Reilly Media, 2011.

[6] Reto Meier. *Professional Android Application Development*. Wiley Publishing, Inc., 2009.

[7] Microsoft. http://msdn.microsoft.com/en-us/library/ms123401, listopad 2012.

[8] Nishith Pathak. *Pro WCF 4: Practical Microsoft SOA Implementation*. Apress, 2011.

[9] Herbert Schild. *C# 4.0 : The Complete Reference*. McGraw-Hill, 2010.