



**POLITECHNIKA  
GDAŃSKA**

WYDZIAŁ ELEKTRONIKI,  
TELEKOMUNIKACJI I INFORMATYKI



Imię i nazwisko studenta: Marcin Olszewski  
Nr albumu: 137357  
Studia pierwszego stopnia  
Forma studiów: stacjonarne  
Kierunek studiów: Informatyka  
Specjalność/profil: -

Imię i nazwisko studenta: Mateusz Pakulski  
Nr albumu: 137359  
Studia pierwszego stopnia  
Forma studiów: stacjonarne  
Kierunek studiów: Informatyka  
Specjalność/profil: -

Imię i nazwisko studenta: PAWEŁ MAZUREK  
Nr albumu: 137342  
Studia pierwszego stopnia  
Forma studiów: stacjonarne  
Kierunek studiów: Informatyka  
Specjalność/profil: -

Imię i nazwisko studenta:  
SEBASTIAN MIAŁKOWSKI  
Nr albumu: 137343  
Studia pierwszego stopnia  
Forma studiów: stacjonarne  
Kierunek studiów: Informatyka  
Specjalność/profil: -

## PRACA DYPLOMOWA INŻYNIERSKA

Tytuł pracy w języku polskim: Asystent treningu biegowego (RunAnd)

Tytuł pracy w języku angielskim: Personal running assistant (RunAnd)

Potwierdzenie przyjęcia pracy	
Opiekun pracy	Kierownik Katedry/Zakładu
<i>podpis</i>	<i>podpis</i>
dr inż. Krzysztof Bruniecki	

Data oddania pracy do dziekanatu:



**POLITECHNIKA  
GDAŃSKA**

WYDZIAŁ ELEKTRONIKI,  
TELEKOMUNIKACJI I INFORMATYKI



## OŚWIADCZENIE

Imię i nazwisko: Marcin Olszewski  
Data i miejsce urodzenia: 14.06.1991, Grudziądz  
Nr albumu: 137357  
Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki  
Kierunek: informatyka  
Poziom studiów: I stopnia - inżynierskie  
Forma studiów: stacjonarne

Ja, niżej podpisany(a), wyrażam zgodę/nie wyrażam zgody\* na korzystanie z mojego projektu dyplomowego zatytułowanego: Asystent treningu biegowego (RunAnd) do celów naukowych lub dydaktycznych.<sup>1</sup>

Gdańsk, dnia .....

.....  
*podpis studenta*

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r., nr 90, poz. 631) i konsekwencji dyscyplinarnych określonych w ustawie Prawo o szkolnictwie wyższym (Dz. U. z 2012 r., poz. 572 z późn. zm.),<sup>2</sup> a także odpowiedzialności cywilno-prawnej oświadczam, że przedkładany projekt dyplomowy został opracowany przeze mnie samodzielnie.

Niniejszy projekt dyplomowy nie był wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. projekcie dyplomowym, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

Potwierdzam zgodność niniejszej wersji projektu dyplomowego z załączoną wersją elektroniczną.

Gdańsk, dnia .....

.....  
*podpis studenta*

Upoważniam Politechnikę Gdańską do umieszczenia ww. projektu dyplomowego w wersji elektronicznej w otwartym, cyfrowym repozytorium instytucjonalnym Politechniki Gdańskiej oraz poddawania jego procesom weryfikacji i ochrony przed przywłaszczeniem jego autorstwa.

Gdańsk, dnia .....

.....  
*podpis studenta*

\*) niepotrzebne skreślić

---

<sup>1</sup> Zarządzenie Rektora Politechniki Gdańskiej nr 34/2009 z 9 listopada 2009 r., załącznik nr 8 do instrukcji archiwalnej PG.

<sup>2</sup> Ustawa z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym:

Art. 214 ustęp 4. W razie podejrzenia popełnienia przez studenta czynu podlegającego na przypisaniu sobie autorstwa istotnego fragmentu lub innych elementów cudzego utworu rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.

Art. 214 ustęp 6. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 4, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o popełnieniu przestępstwa.

## STRESZCZENIE(MARCIN OLSZEWSKI)

Głównym celem Projektu jest implementacja systemu wspomagającego trening biegowy, który będzie składał się z trzech autonomicznych podsystemów, które poprzez wzajemną komunikację będą współtworzyły narzędzie do wspomagania treningu biegowego – RunAnd. Aplikacja dla zawodnika, główny moduł systemu, zaprojektowana zostanie z myślą o użytkownikach systemu Android. Będzie umożliwiała trenującemu wybór trasy do treningu spośród udostępnionych, nawigację podczas biegu z wykorzystaniem map z serwisu Google Maps, wykonywanie i zapisywanie zdjęć, czy przeglądanie archiwum ukończonych treningów. Zawodnik będzie na bieżąco widział, m.in. jaki przebył dystans, jak długo biegnie i ile spalił dotychczas kalorii.

Śledzenie postępów zawodników odbywać się będzie poprzez aplikację Internetową, opartą o *Framework AngularJS*. Trener, oprócz analizy danych (np. aktualnego położenia zawodnika, jego prędkości oraz dystansu, jaki pokonał), będzie mógł w dowolnym czasie wysłać do niego wiadomość, odczytaną po stronie aplikacji mobilnej dzięki rozwiązaniu *TextToSpeech*.

Komunikację pomiędzy wspomnianymi modułami zapewni aplikacja serwerowa zaimplementowana z wykorzystaniem silnika *NodeJS*, wykorzystująca bazę danych *PostgreSQL* oraz udostępniająca usługi sieciowe typu *REST*. Serwer będzie umożliwiał wymianę wiadomości, archiwizację treningów i tras, będzie także przechowywał materiały multimedialne oraz aktualną prognozę pogody.

Ważnym elementem naszego Projektu będą mapy, na których między innymi będzie odbywało się śledzenie postępów w treningu, nawigacja zawodnika oraz projektowanie nowych tras. Skorzystamy z *Google Maps API*, które daje nam szerokie możliwości wykorzystania map. Dzięki temu rozwiązaniu będziemy mieli dostęp do aktualizowanej bazy map z całego świata.

Dodatkowym efektem naszego projektu będzie załącznik w formie instrukcji laboratoryjnej, który pokaże jak szerokie możliwości daje nam wykorzystanie systemów mobilnych. Na przykładzie systemu Android pokażemy, jak wykorzystać programowanie natywne.

Chcielibyśmy, aby system RunAnd przyczynił się do wzrostu(i tak już dużego) zainteresowania biegami, a co za tym idzie motywował jak największą liczbę użytkowników do uprawiania aktywności fizycznej.

### Słowa kluczowe:

nauki techniczne i inżynieryjne, informatyka, Internet, usługi sieciowe, systemy mobilne, geolokalizacja, mapy cyfrowe

## **ABSTRACT(MARCIN OLSZEWSKI)**

## SPIS TREŚCI

Wykaz ważniejszych oznaczeń i skrótów .....	7
1. Wstęp i cel pracy (Marcin Olszewski).....	8
1.1. Cel projektu.....	8
1.2. Motywacje .....	9
1.3. Organizacja pracy i narzędzia wspomagające .....	9
1.3.1. Redmine .....	9
1.3.2. GitHub.....	11
2. Przegląd zagadnień z dziedziny wspomaganie treningu biegowego(Sebastian Miałkowski).....	13
2.1. Przegląd istniejących rozwiązań.....	13
2.1.1. Endomondo .....	13
2.1.2. Runtastic .....	14
2.1.3. MapMyRun .....	15
2.1.4. RunKeeper.....	16
2.2. Opis wykorzystanych algorytmów .....	17
3. Analiza wymagań i projekt funkcjonalny.....	18
3.1. Wizja systemu(Mateusz Pakulski) .....	18
3.1.1. Struktura organizacyjna .....	18
3.1.2. Problemy występujące w organizacji personalnych treningów biegowych .....	18
3.1.3. Udziałowcy systemu .....	18
3.1.4. Cele systemu .....	19
3.1.5. Użytkownicy ich specyfika .....	19
3.2. Aplikacja mobilna .....	21
3.3. Serwer (Paweł Mazurek).....	26
4. Cel .....	<b>Błąd! Nie zdefiniowano zakładki.</b>
5. Komponenty .....	26
6. Wymagania funkcjonalne .....	26
7. Przypadki użycia .....	27
7.1. Aplikacja trenera(Marcin Olszewski) .....	33
8. Architektura systemu oraz przegląd technologii .....	35
8.1. Ogólna architektura całego systemu. ....	35
8.2. Aplikacja mobilna .....	35
<a href="https://github.com/RunandPL/AndroidApp">https://github.com/RunandPL/AndroidApp</a> .....	35
8.3. Serwer (Paweł Mazurek).....	39
<a href="https://github.com/RunandPL/Serwer">https://github.com/RunandPL/Serwer</a> .....	39
Tabela [NR] Limity bazy danych PostgreSQL .....	42
(źródło: <a href="http://www.postgresql.org/about/">http://www.postgresql.org/about/</a> ) .....	42
8.4. Aplikacja trenera (Marcin Olszewski) .....	43
9. Implementacja projektu .....	48
9.1. Aplikacji mobilnej(Sebastian Miałkowski).....	48
Zgodnie z założeniami projektu, aplikacja mobilna zastała zaimplementowana na telefony z systemem Android. Zdecydowaliśmy się na to z dwóch powodów. Pierwszym była chęć lepszego poznania systemu oraz kontynuowanie nauki pisania aplikacji dla niego przeznaczonych, którą rozpoczęliśmy na przedmiocie Programowanie aplikacji mobilnych. Kolejnym, chęć dotarcia z naszą aplikacją do jak największej liczby odbiorców. Biorąc pod uwagę te dwa założenia, wybór systemu Android był wręcz oczywisty, gdyż jak wynika z tabeli [NR TABELI] jest to obecnie najpopularniejszy system na urządzenia mobilne, a jego udział w tym rynku stale rośnie. ....	48
Tabela [NR] Udział systemów mobilnych w rynku .....	48
(źródło: <a href="http://www.idc.com/prodserv/smartphone-os-market-share.jsp">http://www.idc.com/prodserv/smartphone-os-market-share.jsp</a> ).....	48
10. 48	
Dodatkowo zdecydowaliśmy że najniższą wersją systemu wspieraną przez naszą aplikację będzie Android 4.1.2(Jelly Bean). Na podstawie tabeli [NR TABELI] widać że nasza aplikacja będzie wspierać 82.6% urządzeń z systemem firmy Google. ....	49

Tabela [NR] Udział rynkowy poszczególnych wersji systemu Android (źródło: <a href="http://developer.android.com/about/dashboards/index.html">developer.android.com/about/dashboards/index.html</a> ) .....	49
Podczas implementacji aplikacji korzystaliśmy z dwóch frameworków: Butterknife oraz Picasso. Niezbędna również była możliwość wyświetlania map, by to uzyskać zintegrowaliśmy aplikację z serwisem Google Maps.49	
Picasso( <a href="https://square.github.io/picasso/">https:// square.github.io/picasso/</a> ) .....	49
W czasach gdy serwisy społecznościowe cieszą się dużą popularnością, ludzie robią oraz wysyłają na nie coraz więcej zdjęć. Skutkuje to również tym że duża liczba aplikacji musi mieć zaimplementowaną możliwość ich obsługi. Poza tak oczywistymi przykładami jak Facebook, czy Instagram robienie oraz podgląd zdjęć udostępniają aplikacje treningowe. W takich właśnie aplikacjach zastosowanie znajduje framework Picassa. Zdjęcia czy obrazy często mają rozmiary po kilka megabajtów, ich wczytanie więc może powodować blokowanie interfejsu użytkownika. By temu zapobiec takie funkcje można zaimplementować na osobnych wątkach, jednak są to często operacje kłopotliwe oraz występujące w wielu miejscach. By ułatwić sobie pracę można skorzystać z Picasso, który wszystkie operacje wykona za nas a dodatkowo zajmie się obsługą błędów. Framework pozwala wyświetlenie obrazków pobranych z Internetu, wczytanych z dysku oraz pamięci. Prócz samego odczytu plików graficznych została nam udostępniona również możliwość ich prostej edycji np. zmiana rozmiaru, by lepiej mogły się dopasować wyglądu naszej aplikacji. Na wypadek gdyby dostępne możliwości edycji były dla nas niewystarczające, możemy zaimplementować własne i przekazać do wykorzystania dla Picasso. W naszej aplikacji używamy go do pobrania obrazka trasy, który jest generowany przez Google Static Map. Poniżej zamieszczam przykładowy fragment kodu powodujący pobranie obrazka z danego adresu Internetowego, a następnie załadowanie go do widok imageView. ....	49
10.1. Implementacja serwera (Paweł Mazurek) .....	50
10.2. Implementacja aplikacji WWW (Marcin Olszewski) .....	54
11. Rezultaty projektu .....	60
12. Podsumowanie.....	60
12.1. Wnioski .....	60
12.2. Perspektywy.....	60
12.3. Wdrożenie systemu na serwerze produkcyjnym (Paweł Mazurek).....	60
13. W trakcie powstawania projektu, z inicjatywy promotora, zrezygnowaliśmy z wdrożenia systemu na serwerze katedralnym. W ramach zastępstwa, wdrożyliśmy serwer w tym samym środowisku, w którym działa projektowy redmine. To znaczy, na maszynie z systemem Ubuntu 14.04, Node.js w wersji 0.10.31 i serwerem PostgreSQL. Aby restartować serwer, w przypadku aktualizacji plików, lub możliwej awarii, użyliśmy modułu PM2 do Node.js. W przypadku nieoczekiwanego zakończenia pracy procesu, wskrzesza on go. Jeżeli pliki zostaną zaktualizowane, proces serwera jest odnawiany automatycznie, bez okresu przejściowego gdy serwer jest wyłączony. ....	60
14. Wykaz literatury .....	62
15. Załączniki .....	63
15.1. Wykaz ilustracji .....	63
15.3. Wykaz listingów .....	63
15.4. Podział prac.....	63
15.5. Instrukcja instalacji i konfiguracji systemu.....	63
15.6. Instrukcja laboratoryjna (Marcin Olszewski).....	63

## **WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW**

WWW - *World Wide Web*- multimedialny system informatyczny, oparty o hipertekst. Jest jedną z usług udostępnianych w ramach Internetu.

HTTP - *Hypertext Transfer Protocol* - protokół sieci WWW

API - *Application Programming Interface* - ściśle ustalony zestaw reguł komunikacji

URL - *Uniform Resource Locator* - Format adresowania danych wykorzystywany w Internecie i sieciach lokalnych.

POST - Metoda z której pomocą przysyłane są dane w sieci Internet

GET - Sposób w jaki przekazywane są dane pomiędzy dokumentami sieciowymi w sieci http

TTS - *Text-to-Speech* - Multimedialny system głosowego odczytu wiadomości tekstowych.

## 1. WSTĘP I CEL PRACY (MARCIN OLSZEWSKI)

Ważnym elementem życia jest aktywność fizyczna, uprawiana pod wieloma postaciami, na każdym etapie życia. Na temat pozytywnego wpływu aktywności ruchowej napisano już wiele publikacji oraz wykonano szereg badań dotyczących jej wpływu na psychikę człowieka, prawidłowy rozwój fizyczny, badano także wpływ rekreacji na zapadalność na choroby, m.in. miażdżycę, czy infekcje górnych dróg oddechowych. Odnosząc się do jednej z książek na temat aktywności fizycznej, pod tytułem *Sport dla wszystkich*, wydanej przez prof. AWF w Krakowie, dra hab. Ryszarda Winiarskiego, uprawianie aktywności fizycznej pozwala:

- neutralizować stres,
- spowolnić tętno,
- poprawić koordynację nerwowo-mięśniową,
- zwiększyć pojemność życiową płuc,
- zwiększyć nawet 20% objętość krwi, co jednocześnie wpływa na wzrost wydolności organizmu.

Czasem jednak coraz większe tempo życia często powoduje, że rezygnujemy z aktywności fizycznej na rzecz odpoczynku(relaksu biernego). A kiedy mielibyśmy znaleźć czas na planowanie treningów? Wyobraźmy sobie sytuację w której mamy do dyspozycji już gotową bazę treningów przygotowanych przez zawodowych trenerów, na różnych poziomach zaawansowania. Wystarczy zainstalować aplikację i rozpocząć trening. Aplikacja pomoże nam znaleźć właściwą drogę, podpowie jaka może nas spotkać pogoda na trasie, gdzie trenują nasi znajomi, a na koniec pozwoli wysłać innym ciekawe zdjęcie i podzielić się z innymi naszymi podbojami. Nie jest to jednak bujanie w obłokach, tylko rzeczywistość, ponieważ na rynku istnieją już wspomniane rozwiązania. Sytuacja wygląda nieco inaczej, gdybyśmy chcieli trenować nie pod kontrolą aplikacji, ale z udziałem trenera, który na żywo śledzi nasze poczynania i może w każdej chwili przesłać nam wiadomość. Obecność drugiego człowieka, który kontroluje i przypatruje się naszemu treningowi niezwykle pomaga zrealizować cel i motywuje. Aplikację możemy oszukać, a z trenerem spotkamy się twarzą w twarz po zakończonym treningu. Być może stworzymy idealne rozwiązanie dla trenerów biegów przełajowych, rozwiązanie do zdalnego treningu na zróżnicowanym terenie poza miejscem zamieszkania, a może tylko usprawnimy istniejące systemy dla biegaczy. Warto jednak podjąć próbę.

### 1.1. Cel projektu

*Celem projektu jest utworzenie aplikacji mobilnej(Android OS) służącej do wspomagania treningu biegowego za pomocą metod zautomatyzowanych oraz przy udziale trenera monitorującego zdalnie postępy zawodnika.*

Monitoring treningu będzie odbywał się za pomocą strony Internetowej, poprzez którą, trener otrzyma możliwość przesyłania komunikatów do zawodnika, odczytywanych przez TTS (ang.Text-to-Speech). Za pomocą aplikacji Web, trener będzie mógł także zaplanować trening dla swojego zawodnika oraz przeglądać archiwum treningów.



## 1.2. Motywacje

Aktywność fizyczna to niezbędny element zdrowego stylu życia. Od dawna wiadomo także, że bieganie to najprostsza forma aktywności fizycznej, która, jak wspomniano we wstępie, ma korzystny wpływ na prawidłowe funkcjonowanie naszego organizmu. Wpływa również korzystnie na nasze zdolności umysłowe oraz na samopoczucie. Podejmując się realizacji projektu RunAnd, zakładamy, że przyszli użytkownicy systemu otrzymają proste w obsłudze i intuicyjne narzędzie, które pomoże im zaplanować treningi biegowe. Wprowadzając funkcje publikacji tras biegowych, umożliwimy użytkownikom wymianę doświadczeń i osiągnięć. Z drugiej strony chcemy zmierzyć się z systemem od strony technologicznej, używając popularnych technologii mobilnych oraz webowych, odkrywając ich wady i zalety.

## 1.3. Organizacja pracy i narzędzia wspomagające

Odwołując się po raz kolejny do celu naszego projektu, którym niewątpliwie jest wytworzenie systemu wspomagania treningu biegowego, należy wspomnieć także o środkach, które pomogą nam w jego realizacji. Naszym zadaniem nie jest jedynie przygotowanie architektury, analiza wymagań oraz późniejsza implementacja. Ważna jest także droga do osiągnięcia celu, czyli przeprowadzenie projektu informatycznego, w naszym wypadku jest to także projekt inżynierski.

### 1.3.1. Redmine

Na rynku obecnie istnieje kilka czołowych rozwiązań, które oferują nam wsparcie w zarządzaniu projektem (nie tylko, lecz głównie - informatycznym). Należą do nich:

- *JIRA* – zamknięte oprogramowanie australijskiej firmy Atlassian służące do zarządzania projektem oraz śledzenia błędów (tzw. *issue tracker*). Korzystają z niej m.in. programiści Skype. Umożliwia integrację z innymi produktami tej firmy skierowanymi do programistów, np. Confluence,
- *Trac* – projekt open-source napisany w języku Python. Oferuje podobny zakres funkcji, jak JIRA, jednak nastawiony jest na prostotę zarządzania projektem,
- *Redmine* – zaimplementowany w języku Ruby.

Można zauważyć, że przyszłość należy do narzędzi Internetowych. Ponadto wszystkie narzędzia oferują podobną rozpiętość funkcji:

- system zgłaszania i wyszukiwania zadań (ang. *ticket*),
- wsparcie dla priorytetów, statusów oraz przypisania zadań do użytkowników,
- harmonogramowanie zadań,
- wsparcie dla metodyk zwinnych, np. SCRUM,
- integracja z repozytoriami SVN, Git,
- zarządzanie użytkownikami oraz grupami użytkowników,
- fora, dokumenty,
- powiadomienia w obrębie systemu jak i drogą email,
- wykreślenie wykresów Gantt'a, wypalania (ang. *Burn down chart*),
- definiowanie przepływów pracy dla zadań.

Czasem jednak trzeba doinstalować wtyczki(ang. *plugins*), aby móc korzystać z funkcji. Przykładem może być plugin Scrum do Redmine.

#	Typ zagadnienia	Status	Priorytet	Temat	Przypisany do	Data modyfikacji
79	Zadanie	Rozwiązany	Normalny	Działanie serwisu zbierającego informacje o treningu niezależnie od aplikacji.	Sebastian Miałkowski	2014-11-09 00:52
78	Funkcja	Nowy	Niski	Robienie i zapisywanie zdjęć podczas treningu	Sebastian Miałkowski	2014-11-10 15:20
77	Funkcja	Nowy	Normalny	Odczyt wiadomości od trenera TTS	Sebastian Miałkowski	2014-11-09 00:47
76	Funkcja	Nowy	Normalny	Publikacja trasy	Sebastian Miałkowski	2014-11-08 20:15
75	Funkcja	Nowy	Normalny	Akceptacja trenera	Sebastian Miałkowski	2014-11-08 20:14
74	Funkcja	Nowy	Normalny	Wybranie trasy do treningu	Sebastian Miałkowski	2014-11-08 20:12
73	Funkcja	Nowy	Normalny	Wyświetlanie poleceń treningu po zalogowaniu	Sebastian Miałkowski	2014-11-08 20:11
72	Funkcja	Nowy	Wysoki	Opcja treningu dla zalogowanego użytkownika z przypisanym trenerem	Sebastian Miałkowski	2014-11-10 15:20
71	Funkcja	Nowy	Wysoki	Opcja treningu dla zalogowanego użytkownika bez trenera	Sebastian Miałkowski	2014-11-10 15:20
61	Zadanie	Rozwiązany	Normalny	Odczyt głosowy wiadomości od trenera.	Sebastian Miałkowski	2014-11-09 00:47
60	Zadanie	Nowy	Normalny	Wyświetlanie wiadomości od trenera.	Sebastian Miałkowski	2014-11-08 22:04
58	Zadanie	Nowy	Wysoki	Upload bieżącego treningu	Mateusz Pakulski	2014-11-10 15:20
57	Zadanie	Nowy	Normalny	Upload pełnych treningów	Mateusz Pakulski	2014-11-02 20:22
56	Zadanie	Rozwiązany	Normalny	Pominięcie logowania	Mateusz Pakulski	2014-11-02 20:20
50	Zadanie	Rozwiązany	Normalny	Zapis treningu do bazy danych.	Sebastian Miałkowski	2014-11-05 12:15
49	Zadanie	Rozwiązany	Normalny	Dodanie pomiaru czasu przebytej trasy.	Sebastian Miałkowski	2014-11-05 12:15
48	Zadanie	Rozwiązany	Normalny	Pomiar odległości przebytej przez użytkownika trasy.	Sebastian Miałkowski	2014-11-05 12:15
45	Błąd	Rozwiązany	Normalny	Błąd logowania przy rotacji ekranu	Mateusz Pakulski	2014-10-21 21:02
44	Błąd	Rozwiązany	Wysoki	Nie można zalogować się g+ po wylogowaniu	Mateusz Pakulski	2014-10-19 12:00
43	Zadanie	Rozwiązany	Normalny	Wylogowywanie	Mateusz Pakulski	2014-10-19 00:38

Rysunek 1.1. Lista zagadnień w systemie Redmine

Zdecydowaliśmy się na system Redmine, ponieważ został już wcześniej wypróbowany przez nas nie tylko w projektach informatycznych. Jest prosty w obsłudze i w bardzo prosty sposób pozwala przygotowywać tematy(ang. *templates*) poprzez nadpisywanie głównego arkusza stylów CSS oraz opcjonalne dodanie kodu JavaScript. Dla ułatwienia wykorzystaliśmy kod CSS do przydzielenia kolorów priorytetom zgłoszeń. Nasz system dostępny jest pod adresem <http://runand.greeters.pl>.

Kolejną wprowadzoną przez nas zmianą było wprowadzenie ustalenie nowych typów zagadnień(ang. *issue*) i zdefiniowanie dla nich przepływów pracy. Ponadto, system redmine pozwala w skonfigurować na wiele sposobów opcje projektu. Począwszy od wyboru modułów dostępnych w projekcie(m.in. pliki, wiki, fora, kalendarz), szerokiej gamy uprawnień dla poszczególnych użytkowników oraz całych grup, definiowanie niestandardowych grup użytkowników, dodatkowych pól przy tworzeniu zagadnień, czy tworzenia kwerend przechowujących informacje o sposobie wyświetlania listy zagadnień.

## Przepływ pracy

Kopia Podsumowanie

Przejścia między statusami Uprawnienia do pól

Zaznacz rolę i typ zagadnienia do edycji przepływu pracy:

Rola: Kierownik Typ zagadnienia: Funkcja Edytuj Wyświetlaj tylko statusy używane przez ten typ zagadnienia

Obecny status	Uprawnione nowe statusy					
	Nowy	W toku	Rozwiązany	Odpowiedź	Zamknięty	Odrzucony
Nowy	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
W toku	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Rozwiązany	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Odpowiedź	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Zamknięty	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Odrzucony	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Additional transitions allowed when the user is the author  
Additional transitions allowed when the user is the assignee

Zapisz

Rysunek 1.2. Definiowanie przepływu pracy w systemie Redmine

Wspomniane typy zagadnień to:

- funkcja** – określa funkcję systemu z perspektywy użytkownika, np. *Wysłanie wiadomości od trenera w trakcie treningu*,
- zadanie** – czynność niezbędna do implementacji funkcji, np. *Integracja z Google Maps API*,
- błąd** – zgłoszenie znalezione przy implementacji błędu, np. *Błąd logowania przy rotacji ekranu*,
- rozdział** – zagadnienie związane z dokumentacją, np. *Spis treści*.

### 1.3.2. GitHub

„Nie trzeba nikogo przekonywać, że współczesny twórca nie podejmuje pracy nad żadnym projektem nie mając jakiejś strategii tworzenia kopii zapasowej swojej pracy.” Zgodnie ze zdaniem Pana Joe Loeliger i Matthew McCullough, którzy napisali książkę pt. *Kontrola wersji z systemem Git*, zdecydowaliśmy się na wykorzystanie repozytorium Git (rozproszonego systemu kontroli wersji) w naszej pracy. Zdecydowaliśmy się skorzystać w tym celu z serwisu GitHub, który oferuje programistom darmowy hosting programów open source. Po utworzeniu konta w serwisie oraz utworzenia publicznych projektów (tworzenie prywatnych repozytoriów jest możliwe, ale płatne), wystarczy znać kilka podstawowych komend do pracy z naszym repozytorium.

```
1. touch README.md
2. git init
3. git add README.md
4. git commit -m "first commit"
5. git remote add origin https://github.com/RunandPL/ExampleRunAnd.git
6. git push -u origin master
7.
8. git remote add origin https://github.com/RunandPL/ExampleRunAnd.git
```

```
9. git push -u origin master
10.
11. git pull origin master
12.
13. git config user.name „Marcin Olszewski”
14. git config user.email "marolsze@student.pg.gda.pl"
```

Pierwszy sposób (linie 1-5 listingu) polega na utworzeniu repozytorium i jego struktury (folder .git) w katalogu projektu (linia 2) oraz powiązaniu go z naszym projektem w serwisie GitHub i wysłaniu pierwszego zapytania typu push, dzięki któremu dodamy nasze zmiany potwierdzone poleceniem git commit do naszego projektu. Drugi sposób zakłada, że posiadamy już katalog z projektem, w którym mamy już nasze repozytorium i chcemy np. wysłać je do pustego projektu w serwisie GitHub. Wtedy nie trzeba wykonywać polecenia *git init*.

Jako, że jedną z zalet Gita jest możliwość współdzielenia repozytorium przez wielu użytkowników, powinniśmy zawsze przed wykonaniem komendy git push, pobrać aktualną wersję projektu poleceniem git pull (linia 10). Jeśli aktualna wersja mocno różni się od naszej i nie będzie możliwe automatyczne połączenie wersji (ang. *merge*), zostaniemy poproszeni o ręczne usunięcie konfliktów. Użyteczne mogą okazać się komendy z linii 12 i 13 służące do konfiguracji naszej tożsamości w ramach repozytorium.

Aby móc wykonywać powyższe komendy w środowisku Windows, musimy zainstalować np. program msysGit lub Cygwin. Użytkownicy systemów z rodziny Unix, mają uproszczone zadanie. W dystrybucji systemu Linux Ubuntu, wystarczy w wierszu poleceń wpisać `$ apt-get install git`.

*Repozytorium projektu dostępne jest pod adresem <https://github.com/RunandPL>.*

## **2. PRZEGLĄD ZAGADNIEŃ Z DZIEDZINY WSPOMAGANIA TRENINGU BIEGOWEGO(SEBASTIAN MIAŁKOWSKI)**

### **2.1. Przegląd istniejących rozwiązań**

#### *2.1.1. Endomondo*

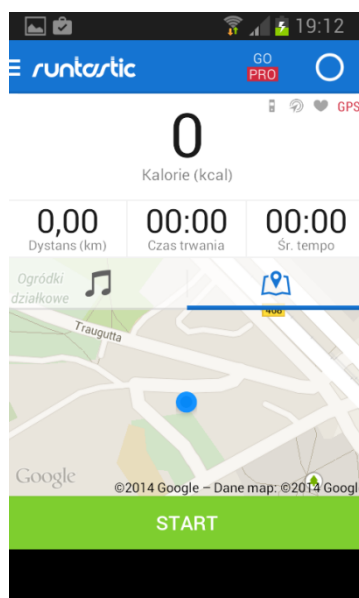
Jest obecnie jedną z najpopularniejszych lub nawet najpopularniejszą aplikacją wspomagającą treningi. Jest to bardzo rozbudowane narzędzie, umożliwiające rejestrowanie treningów w 58 różnych dyscyplinach. Podczas każdego z nich mamy możliwość rejestrowania podstawowych statystyk każdego z nich. Endomondo wspiera również funkcję TTS (TextToSpeech). Dzięki temu aplikacja może w trakcie treningu informować nas o jego statystykach. Na popularność tej aplikacji nie wątpliwie wpływa jej uniwersalność, co może również być jej największym minusem. Według mnie autorzy tej aplikacji niepotrzebnie dodawali możliwość rejestracji kilku rodzajów aktywności. Jako przykład mogą posłużyć sztuki walki. Nie trzeba nikogo przekonywać że są one sportem mocno kontaktowym oraz dynamicznym. Z tych powodów nie ma możliwości by podczas ich uprawiania mieć przy sobie telefon. Należy również pamiętać że każdy taki trening może wyglądać inaczej, przez co wskazania aplikacji np. odnośnie spalonych kalorii będą z pewnością mocno przekłamane. Pomimo kilku minusów jest to aplikacja bardzo udana, posiadająca dużo ciekawych i przydatnych funkcji. Pierwszą z nich, obecną w większości dostępnych na rynku aplikacji, jest możliwość podążania wcześniej wytyczoną trasą. Każdy użytkownik aplikacji, ma możliwość poprzez stronę Internetową stworzenia trasy, którą może następnie udostępnić. Tak udostępnione trasy stają się dostępne dla wszystkich użytkowników, którzy mogą na nich następnie trenować. Daje to możliwość rywalizacji oraz dzielenia się ciekawymi miejscami do uprawiania sportu. Problemem jest w tym wypadku brak sprawdzania położenia użytkownika względem trasy. W czasie biegu, kiedy nie patrzymy się na ekran telefonu aplikacja nie powie nam kiedy mam skręcić lub czy dobiegliśmy do końca. Mówiąc o rywalizacji, należy wspomnieć o kolejnej ciekawej funkcji. Nazywa się ona po prostu Rywalizacje. Co miesiąc w aplikacji pojawiają się konkurencje polegające osiągnięciu ustalonego celu np. przebiegnięciu największej liczby kilometrów, spaleniu największej liczby kalorii. Każda konkurencja trwa kilka dni. Na koniec osoby z najwyższymi wynikami wygrywają nagrody, wartość jak również ich ilość zależy od konkurencji gdyż niektóre z konkurencji są sponsorowane przez zewnętrzne firmy. Powyższe funkcje dostępne są w wersji darmowej aplikacji, istnieje możliwość kupienia jej wersji premium w której uzyskujemy dostęp do dodatkowych możliwości. Wśród nich znajdują się personalne treningi, strefy tętna, krokomierz, informacje o pogodzie w czasie treningu oraz edytowalny trener audio. Niestety nie ma możliwości przetestowania tych funkcji nie posiadając płatnej wersji, przydałaby się możliwość kilku dniowego testowania. Sam wygląd aplikacji, przedstawiony na 2.1., jest w mojej opinii bardzo dobrze zrobiony, menu jest przejrzyste i czytelne a wszystkie opcje dobrze rozplanowane. Również strona Internetowa aplikacji jest intuicyjna oraz wygląda przyjemnie. Nawet odwiedzając ją pierwszy raz bez kłopotu znajdziemy szukane opcje.



2.1. Ekran treningu aplikacji Endomondo

### 2.1.2. Runtastic

Jest kolejną aplikacją treningową dostępną na urządzenia mobilne. Tak samo jak Endomondo, wspiera na wiele typów dyscyplin. Również tutaj mamy możliwość pobrania jej bezpłatnej wersji, jak również zakupienia opcji Pro. W ramach bezpłatnej wersji istnieje możliwość odbycia zwykłego treningu lub z ustalonymi wcześniej przez nas celami, odległością lub czasem. Każda z odbytych przez nas aktywności jest zapisywana, co daje możliwość ich późniejszego podglądu oraz określenia naszych postępów. Runtastic udostępnia również możliwość podglądu statystyk naszych aktywności z ostatnich dwóch miesięcy, w wersji Pro również z ostatniego tygodnia oraz roku. Wykupienie wersji Pro daje oczywiście dostęp do dodatkowych funkcji. Pierwszą z nich są trasy, posiadają one działanie identyczne jak te w Endomondo. Również tutaj powinny być lepiej zaimplementowane. Wybierając trening po trasie nie jest sprawdzane nasze położenie, jak również nie informacji głosowych informujących o kierunku trasy. Kolejną opcją są treningi interwałowe, podczas takiego treningu dostajemy informacje głosowe o kolejnych interwałach. Wersja Pro udostępnia również opcję śledzenia na żywo, dzięki czemu nasza pozycja jest na bieżąco aktualizowana i wysyłana na serwer. Za pomocą strony Internetowej inni użytkownicy mają potem możliwość podglądania naszej pozycji oraz innych statystyk naszego treningu. Gdy zauważą że nasze tempo spada i będą chcieli nas zmotywować mogą nas o tym powiadomić poprzez wysłanie wiadomości, która zostanie dla nas głosowo odczytana. Jedną z najciekawszych funkcji umilających trening są biegi fabularne, jednak jest to opcja dodatkowo płatna. Podczas biegu fabularnego aplikacja opowiada nam jedną z historii, którą wybraliśmy. Jest ona opowiadana w sposób taki że czujemy się jej częścią co dodatkowo nas motywuje. Dodatkowo wykupić możemy również plany treningowe, ułożone przez prawdziwych trenerów. Twórcy aplikacji mogliby udostępnić trening przykładowy w celu zapoznania się z działaniem tej funkcji. Runtastic, jak widać na 2.2., posiada bardzo dobrze rozplanowany i ładny wygląd. Dostęp do wszystkich okien aplikacji odbywa się poprzez boczną szufladę akcji. Każdy, nawet początkujący użytkownik tej aplikacji bez problemu się w niej odnajdzie.

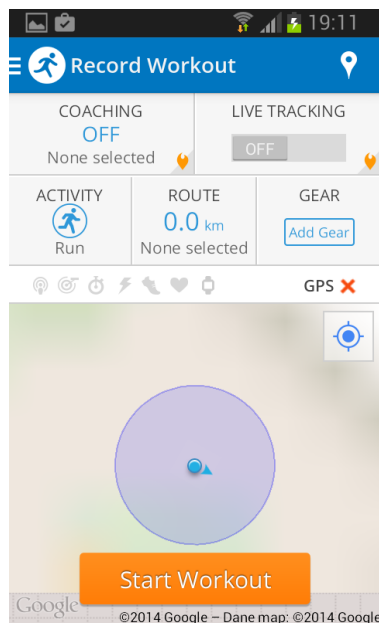


2.2. Ekran treningu aplikacji Runastic

### 2.1.3. MapMyRun

Kolejna z aplikacji przeznaczonych dla sportowców, która jest przy tym najbardziej rozbudowana. Do wyboru mamy w niej największą liczbę dyscyplin treningowych, wśród których znajdują się np. wyprowadzanie psa, śpiewanie, skok o linie czy rąbanie drewna. Prócz funkcji związanych z treningiem, aplikacja daje nam dodatkowo możliwość rejestrowania zjedzonych posiłków. To w połączeniu z informacjami o treningu pozwala na kontrole naszego bilansu kalorycznego i zbilansowanie diety. W informacjach o profilu dostępne są informacje o średnim tygodniowym przebytnym dystansie, czasie treningów, ich ilości oraz spalonych kaloriach. Takie same informacje dostępne są również całej historii treningowej. Z funkcji dostępnych w darmowej wersji możemy wybrać tylko dwa rodzaje treningu, zwykły oraz po trasie. Dodatkowo w ich czasie możemy robić zdjęcia. Jest to mocno ograniczony wybór w porównaniu do poprzednich aplikacji gdzie istnieje możliwość ustalenia odległości jaką chcemy pokonać czy też czasu jaki chcemy trenować. Zastępstwem tych opcji może być funkcja udostępniona na stronie Internetowej aplikacji. Nosi ona nazwę „Stwórz Cel” i polega na ustaleniu dla siebie zadania do wykonania. Takim zadaniem może być np. przebiegnięcie 100 mil w ciągu 4 tygodni. Tym co może zachęcić do korzystania z tej aplikacji są funkcje niedostępne w innych, lista wydarzeń sportowych oraz informacje o zużyciu obuwia. Pierwsza z nich pozwala nam na łatwe śledzenie informacji o imprezach, które będą miały miejsce w przyszłości w naszej okolicy i przygotowanie się do nich. W treningu biegowym równie ważne jak dobra technika są też dobre buty, które jednak się zużywają. Jednak kontrola kilometrów do czasu wymiany obuwia może być problematyczna, z tego względu funkcja dostępna w MapMyRun jest niezwykle pomocna. W wersji premium, która w wypadku tej aplikacji nosi nazwę MVP dostajemy dostęp do treningów interwałowych, śledzenia na żywo, analizy pracy serca, planów treningowych oraz trenera audio. Należy zwrócić uwagę że liczba funkcji jest ograniczona w porównaniu do innych aplikacji. Na duży plus należy ocenić wygląd, przedstawiony na rysunku [NUMER RYSUNKU], oraz funkcjonalność aplikacji. Jest ona przyjemna dla oka, poszczególne okna są dobrze rozplanowane więc nie ma

żadnego problemu z odnalezieniem funkcji jakiej potrzebujemy. Identyczne wrażenie można odnieść korzystając ze strony Internetowej, jest ona utrzymana w takiej samej stylistyce i równie przyjemna w użytkowaniu.



2.3. Ekran treningu aplikacji MyMapRun

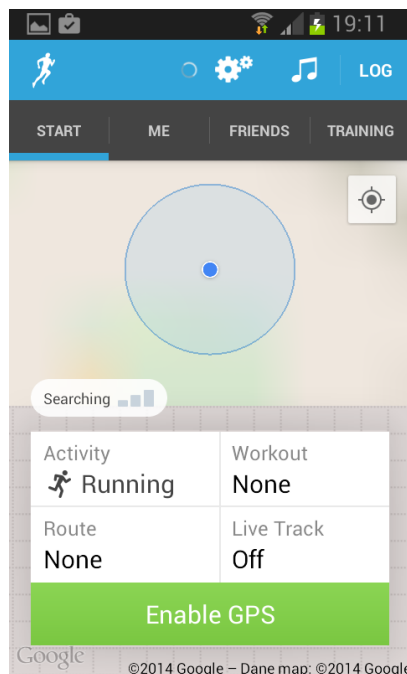
#### 2.1.4. RunKeeper

Ostatnią z omawianych przeze mnie aplikacji jest RunKeeper stworzona przez firmę o tej samej nazwie. W porównaniu do wymienianych wcześniej aplikacji charakteryzuje się ona największą liczbą dostępnych opcji w darmowej wersji. Bardzo ciekawie przedstawia się wybór aktywności, które są podzielone na dwie kategorie, GPS oraz stoper. Każda z nich zawiera aktywności wykonywane przy pomocy odpowiedniego narzędzia. Dla przykładu, bieganie zostało przydzielone do obu kategorii a CrossFit do stopera. Wybór przez nas rodzaju treningu ma wpływ na wygląd głównego ekranu aplikacji po jego rozpoczęciu. Podczas treningu z kategorii GPS na głównym ekranie mamy wyświetlone podstawowe informacje np. czas, średnio tempo. Prócz danych znajdują się tam dodatkowo przyciski służące za modyfikację interfejsu np. przełączenie go w tryb nocny, oraz robienie zdjęć. W przypadku aktywności z kategorii stoper na ekranie jest wyświetlana tarcza zegara, ponad nią czas treningu a poniżej przyciski „Stop” oraz „Pauza”. Całość jest bardzo czytelna dzięki czemu czas naszego ćwiczenia można śledzić z odległości kilku metrów. Przeglądając historię ćwiczeń mamy dostęp do wykresów przedstawiających m.in. średni tempo, wysokość na poziomie morza oraz puls. Tej opcji zdecydowanie brakuje w darmowych produktach konkurencji. Dodatkowo można zobaczyć tempo w jakim pokonywaliśmy poszczególne kilometry. Również informacje audio są bardzo rozbudowane, jeżeli tego zachcemy mamy możliwość otrzymywania informacji o prędkości, tempie, odległości czy pulsie. Poprzednie aplikacje miały tę funkcję dużo uboższą. Wersja premium, RunKeeper Elite, daje dostęp do takich samych funkcji jak w przypadku poprzednich aplikacji. Są to plany treningowe, śledzenie na żywo oraz jeszcze bardziej rozbudowane statystyki. Jednak RunKeeper posiada też kilka mniejszych i większych minusów. Tym co może przeszkadzać



początkującemu użytkownikowi jest wygląd aplikacji, pokazany na rysunku [NUMER RYSUNKU], oraz rozmieszczenie w niej funkcji.

Sam mając pierwszy raz styczność z tą aplikacją czułem się lekko zagubiony, uczucie to minęło dopiero po kilku minutach obcowania z nią. Duże większym problemem, który ujawnia się dopiero po jakimś czasie jest jej zacinanie się. Wiele razy, podczas normalnego użytku, natrafiałem na chwilowe zawieszenia się lub pojawiania czarnego ekranu. Mimo że nie trwały długo w dużym stopniu psuły przyjemność użytkowania.



2.4. Ekran aplikacji RunKeeper

## 2.2. Opis wykorzystanych algorytmów

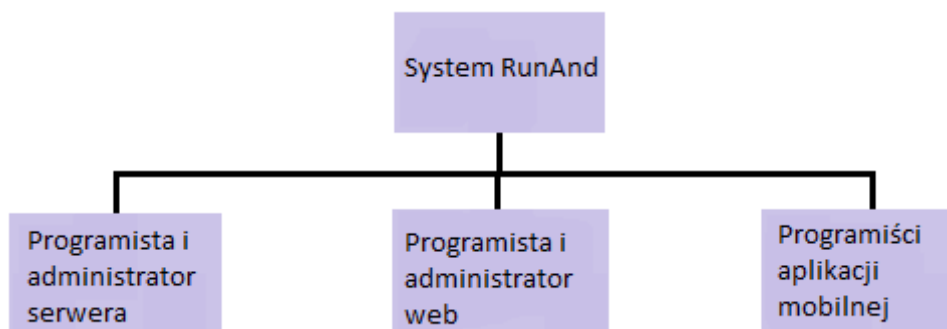
Opis wykorzystanych algorytmów powinien znaleźć się tutaj.

### 3. ANALIZA WYMAGAŃ I PROJEKT FUNKCJONALNY

#### 3.1. Wizja systemu(Mateusz Pakulski)

System skierowany jest dla trenerów osobistych prowadzących zawodników przygotowujących się do startów w zawodach biegowych na długich i średnich dystansach oraz do indywidualnych zawodników.

##### 3.1.1. Struktura organizacyjna



3.1. Struktura organizacyjna

Tabela 3.1. Odpowiedzialność jednostek

Jednostka organizacyjna	Zakres odpowiedzialności
Programista i administrator serwera	Zapewnienie poprawnego działania serwerów i naprawa ewentualnych błędów.
Programista i administrator web	Zapewnienie poprawnego działania aplikacji webowej, naprawa błędów, udoskonalanie UI.
Programiści administracji mobilnej	Konserwacja, wymiana Serwera danych oraz stacji roboczych

##### 3.1.2. Problemy występujące w organizacji personalnych treningów biegowych

Do zidentyfikowanych problemów należy zaliczyć:

- utrudnioną komunikację zawodnika z trenerem,
- trener może monitorować postęp zawodnika tylko po zakończonym treningu,
- trener nie może zmienić zaleceń treningowych w trakcie trwania treningu w zależności od wyników zawodnika,
- brak możliwości przeprowadzania treningu z kilkoma zawodnikami w tym samym czasie.

##### 3.1.3. Udziałowcy systemu

Tabela 3.2. Udziałowcy systemu

Udziałowiec	Punkt widzenia
Właściciel	Nadzieje: <ul style="list-style-type: none"><li>• monopolizacja rynku w dziedzinie aplikacji,</li><li>• ułatwiających pracę zawodnika z trenerem.</li></ul> Obawy: <ul style="list-style-type: none"><li>• problemy z działaniem serwerów i aplikacji,</li><li>• wyciek danych.</li></ul>

Udziałowiec	Punkt widzenia
Trener	<p>Nadzieje:</p> <ul style="list-style-type: none"> <li>• zwiększenie komfortu pracy,</li> <li>• usystematyzowanie rekordów pacjentów.</li> </ul> <p>Obawy:</p> <ul style="list-style-type: none"> <li>• błędy w praktyce wynikające z działania systemu,</li> <li>• zagubienie/usunięcie akt pacjentów.</li> </ul>
Zawodnik	<p>Nadzieje:</p> <ul style="list-style-type: none"> <li>• poprawa jakości treningów,</li> <li>• lepsza komunikacja z trenerem.</li> </ul> <p>Obawy:</p> <ul style="list-style-type: none"> <li>• niepotrzebne skomplikowanie treningów,</li> <li>• ryzyko awarii systemu w trakcie treningu i pozostanie bez instrukcji.</li> </ul>
Administratorzy/Programiści	<p>Nadzieje:</p> <ul style="list-style-type: none"> <li>• mała awaryjność systemu</li> </ul> <p>Obawy:</p> <ul style="list-style-type: none"> <li>• duża ilość pracy związana z awaryjnością systemu</li> </ul>

### 3.1.4. Cele systemu

Tabela 3.3. Cele systemu

Cel	Mierzalne kryterium
Poprawa jakości treningów	Lepsza komunikacja trenera z zawodnikiem, szybszy postęp zawodnika, lepiej dopasowany plan treningowy
Zwiększenie zysków trenerów	Wyższe zarobki trenerów poprzez pracę z kilkoma zawodnikami w tym samym czasie
Zdalna komunikacja z zawodnikiem	Mniejszy stopień zmęczenia trenera po treningu, większa wygoda pracy trenera i zawodników

### 3.1.5. Użytkownicy i ich specyfika

Tabela 3.4. Użytkownicy i ich specyfika

Użytkownik	Specyfika	Opis specyfiki
Recepcjonista	Umiejętności obsługi systemów IT, potrzeba pomocy	Średnia umiejętność obsługi systemów IT.
	Specyficzne warunki pracy, najważniejsze aspekty pracy	brak
	Wymagania dotyczące interfejsu użytkownika	Przejrzysty interfejs desktopowy z możliwością podglądu terminów wizyt
Administratorzy	Umiejętności obsługi systemów IT, potrzeba pomocy	Znakomita znajomość systemu.
	Specyficzne warunki pracy, najważniejsze aspekty pracy	Możliwość dostępu zdalnego do systemu.
	Wymagania dotyczące interfejsu użytkownika	Brak wymagań
Trener	Umiejętności obsługi systemów IT, potrzeba pomocy	Słaba lub umiarkowana umiejętność obsługi systemów IT.
	Specyficzne warunki pracy, najważniejsze aspekty pracy	Przy biurku, lub w dowolnym miejscu przy użyciu urządzenia przenośnego z dostępem do Internetu.

Użytkownik	Specyfika	Opis specyfiki
	Wymagania dotyczące interfejsu użytkownika	Graficzny przejrzysty interfejs z intuicyjną obsługą.

### 3.1.6. Inne systemy i ich interfejsy

Tabela 3.5. Inne systemy i ich interfejsy

System współpracujący	Interfejs (udostępniane / wywoływane funkcje, transmitowane dane)
Logowanie Google+	Autoryzacja użytkownika do systemu za pomocą interfejsu Google OAuth 2.0
Mapy Google	Wyświetlanie map za pomocą interfejsu Google Maps 2.0

### 3.1.7. Wymagania jakościowe

Tabela 3.6. Wymagania jakościowe

Kategoria	Treść wymagania	Priorytet
wydajność	Duże wymagania wynikające z konieczności obsługi wielu klientów naraz. Trening może być przeprowadzony przez wielu trenerów równocześnie z wieloma zawodnikami. Dodatkowo system umożliwia treningi biegaczom nieprzypisanym do konkretnego trenera, których wyniki również muszą być aktualizowane.	4
niezawodność	System musi być niezawodny. Każde awarie systemu powodować będą niemożliwość realizacji treningów i zapisu postępów.	3
dostępność	System powinien pracować przez cały czas w ciągu doby, ewentualne backoupy powinny odbywać się w równoległe, aby nie zakłócić treningów. Przerwy w działaniu systemu spowodowane awarią powinny być niezwłocznie naprawione.	1
ochrona	System musi być odporny na ataki z zewnątrz ponieważ będzie przechowywał dane biegaczy, oraz trenerów. Jednakowoż nie będą to dane krytyczne.	5
przenośność	Brak wymagań dotyczących przenośności systemu pomiędzy różnymi systemami operacyjnymi, jednakowoż system rejestracji online powinien działać poprawnie na wszystkich popularnych przeglądarkach Internetowych.	7
elastyczność	System musi udostępniać możliwość łatwego wprowadzania zmian wynikających ze zmiany przepisów.	8
konfigurowalność	System musi udostępnić możliwość zmiany parametrów konfiguracyjnych	6
interfejs użytkownika	Powinien umożliwiać jak najszybszą pracę z systemem, być jak najbardziej intuicyjny, jednak powinien umożliwiać również możliwość zmiany kluczowych parametrów.	2

### 3.1.8. Ograniczenia dotyczące zasobów projektowych wykonawcy

Czasowe: 09.12.2014 r.

Budżetowe: Budżet własny

Ludzkie: 4 programistów

Sprzętowe:

- maszyna serwerowa,
- 4 Komputery PC,
- 2 Urządzenia przenośne z Systemem Operacyjnym Android w wersji 4.0 lub wyższej

Oprogramowanie: Oprogramowanie bezpłatne

Ze względu na planowane wdrożenie na serwerze katedralnym komunikacja z serwerem odbywać się musi wyłącznie po protokole HTTP.

### 3.1.9. Ograniczenia dotyczące produktu

Zawodnik musi mieć dostęp do systemu z urządzenia mobilnego niezależnie od miejsca przebywania.

Określony sprzęt:

- zawodnik - urządzenie przenośne z Systemem Operacyjnym Android w wersji 4.0 lub wyższej,
- trener - komputer PC lub urządzenie mobilne z dostępem do Internetu i przeglądarką WWW.

Narzucona technologia wykonania:

- serwer - NodeJS
- aplikacja webowa – AngularJS, Bootstrap
- aplikacja mobilna - Android

Określone formaty danych:

- serwer - baza danych PostgreSQL
- aplikacja mobilna - baza danych SQLite
- wymiana informacji - pliki JSON

Wymaganą dokumentację stanowi praca inżynierska. Natomiast wdrożenie będzie obejmowało uruchomienie serwera i aplikacji webowej oraz udostępnienie aplikacji mobilnej dla użytkowników. Istnieje możliwość wdrożenia systemu na serwerze katedralnym.

### 3.2. Aplikacja mobilna

Aplikacja Mobilna ma na celu udostępnienie zawodnikom możliwość rejestrowania danych gromadzonych podczas treningu, wyświetlenia ich do późniejszej analizy, a także ułatwić treningi personalne nadzorowane przez trenera. Za pośrednictwem serwera dane treningowe mogą zostać wysłane w czasie rzeczywistym, lub po zakończonym treningu do trenera nadzorującego treningi. W odpowiedzi na nadesłane dane trener może przekazać informacje do zawodnika. Wiadomości otrzymane od trenera odczytane zostaną za pośrednictwem systemu TTS. Poza przesyłaniem danych biometrycznych i lokalizacyjnych zawodnik ma możliwość robienia zdjęć podczas treningu. Zdjęcia zrobione podczas treningu przekazane zostają na serwer wraz z zapisem całej sesji treningowej bezpośrednio po jej zakończeniu.

#### 3.2.1. Pakiety

Aplikacja mobilna składać się będzie z pakietów klas języka Java podzielonych według funkcjonalności. Rozróżnione będą pakiety odpowiedzialne za komunikację z serwerem, przeprowadzenie treningu, przechowywanie danych, serwisy lokalizacyjne, oraz komunikację z bazą danych.

Tabela 3.7. implementowane funkcje aplikacji mobilnej

Redmine ID	Priorytet	Temat	Uwagi
78	Niski	Robienie i zapisywanie zdjęć podczas treningu	Możliwość robienia zdjęć podczas treningu, zapisywania ich, a także geotagowania.
77	Wysoki	Odczyt wiadomości od trenera TTS	
76	Wysoki	Publikacja trasy	Należy rozróżnić trasy prywatne od tras



Na diagramie celem zwiększenia czytelności nie zostały przedstawione przypadki użycia odpowiedzialne za komunikację z serwerem. Reprezentowane są one przez połączenie poprzedzającego je przypadku użycia z aktorem serwer. Przypadki wymagające komunikacji z serwerem, a niepołączone z aktorem serwer to *logowanie* i *rejestracja*. Opisane zostały tylko najważniejsze przypadki użycia. Większość przypadków użycia wymaga uprawnień zalogowanego użytkownika co odnotowane zostało w ich opisach.

Tabela 3.8. Logowanie Google+

Nazwa przypadku użycia	Logowanie G+
Zdarzenie wywołujące	Przy uruchomieniu aplikacji użytkownik klika czerwony przycisk „Zaloguj się”
Opis	Obsługuje logowanie użytkownika za pomocą konta Google+
Aktorzy	Użytkownik, Serwer, Serwer Google
Warunki początkowe	Brak zalogowania
Przebieg	<ol style="list-style-type: none"> <li>1. Wyświetlony zostaje ekran uprawnień.</li> <li>2. Użytkownik akceptuje regulamin.</li> <li>3. Wyświetlony zostaje Progress Dialog do czasu zakończenia procesu uwierzytelniania.</li> <li>4. Następuje przekierowanie do głównej aktywności.</li> </ol>
Warunki końcowe	Użytkownik otrzymuje z serwera wiadomość potwierdzającą wykonanie operacji.
Sytuacje wyjątkowe	<ol style="list-style-type: none"> <li>1. Błąd połączenia z serwerem Google.</li> <li>2. Błędne dane konta Google.</li> <li>3. Brak dostępu do Internetu.</li> <li>4. Otrzymanie kodu błędu z serwera aplikacji</li> </ol>
Rozwiązywanie sytuacji wyjątkowych	Wyświetlenie użytkownikowi komunikatu. W przypadku błędnych danych Aplikacja przekierowuje użytkownika do widoku webowego, w którym może on dokonać zmiany loginu i hasła.

Tabela 3.9. Rejestracja użytkownika

Nazwa przypadku użycia	Rejestracja standardowego użytkownika
Zdarzenie wywołujące	Użytkownik klika opcję „Zarejestruj” znajdującą się na w dolnej części ekranu logowania.
Opis	Obsługuje rejestrację nowego użytkownika systemu, nieposiadającego konta Google+.
Aktorzy	Użytkownik, Serwer
Warunki początkowe	Brak zalogowania. Brak utworzonego konta.
Przebieg	<ol style="list-style-type: none"> <li>1. Użytkownik wypełnia formularz rejestracyjny podając swój adres email i nowe hasło.</li> <li>2. Użytkownik klika przycisk „Zarejestruj”.</li> <li>3. Przeprowadzana zostaje wstępna walidacja.</li> <li>4. Wysyłane jest do serwera zapytanie rejestrujące nowego użytkownika.</li> <li>5. Wyświetlony zostaje komunikat o wyniku operacji i użytkownik zostaje przekierowany do aktywności logowania.</li> </ol>
Warunki końcowe	Użytkownik otrzymuje z serwera wiadomość potwierdzającą wykonanie operacji.
Sytuacje wyjątkowe	<ol style="list-style-type: none"> <li>1. Niepoprawnie wypełniony formularz.</li> <li>2. Otrzymanie z serwera kodu błędu.</li> <li>3. Brak dostępu do Internetu.</li> </ol>
Rozwiązywanie sytuacji wyjątkowych	Wyświetlenie użytkownikowi komunikatu (W przypadku błędu walidacji również dokładny opis). W przypadku wygaśnięcia tokenu autoryzacyjnego następuje przekierowanie do aktywności logowania.

Tabela 3.10. Dodanie trenera

Nazwa przypadku użycia	Dodanie/Odrzucenie trenera
Zdarzenie wywołujące	Użytkownik klika w opcję menu „Dodaj trenera”
Opis	Obsługuje przypisanie bądź odrzucenie trenera, który wysłał do zawodnika zaproszenie.
Aktorzy	Użytkownik, Serwer
Warunki początkowe	Uprawnienia zalogowanego użytkownika, otrzymane zaproszenie, połączenie z Internetem
Przebieg	<ol style="list-style-type: none"> <li>1. Wysyłane jest do serwera zapytanie mające na celu pobranie</li> </ol>

	zaproszeń oczekujących na rozpatrzenie. 2. Po otrzymaniu odpowiedzi z serwera widok jest aktualizowany. 3. Użytkownik otwiera menu kontekstowe dla wybranego trenera poprzez przytrzymanie elementu listy który go reprezentuje. 4. Użytkownik wybiera akcję dodaj lub odrzuć. 5. Wysyłane zostaje na serwer zapytanie mające na celu zrealizowanie wybranej czynności. 6. Użytkownik zostaje poinformowany o wyniku operacji.
<b>Warunki końcowe</b>	Użytkownik otrzymuje z serwera wiadomość potwierdzającą wykonanie operacji. W przypadku operacji dodawania zostaje przekierowany do aktywności głównej, natomiast w przypadku odrzucenia użytkownik pozostaje w aktywności dodawania trenera celem podjęcia dalszych działań.
<b>Sytuacje wyjątkowe</b>	1. Otrzymanie z serwera kodu błędu. 2. Brak dostępu do Internetu.
<b>Rozwiązywanie sytuacji wyjątkowych</b>	Wyświetlenie użytkownikowi komunikatu. W przypadku wygaśnięcia tokenu autoryzacyjnego następuje przekierowanie do aktywności logowania.

Tabela 3.11. Zmiana hasła

<b>Nazwa przypadku użycia</b>	<b>Zmiana/Ustawienie hasła</b>
<b>Zdarzenie wywołujące</b>	Użytkownik klika w opcje menu „Moje dane”
<b>Opis</b>	Obsługuje przypisanie(dla użytkowników Google+) lub zmianę hasła użytkownika.
<b>Aktorzy</b>	Użytkownik, Serwer
<b>Warunki początkowe</b>	Uprawnienia zalogowanego użytkownika, połączenie z Internetem
<b>Przebieg</b>	1. Wysyłane jest do serwera zapytanie mające na celu pobranie danych użytkownika. 2. Po otrzymaniu odpowiedzi z serwera widok jest aktualizowany. 3. Użytkownik podaje nowe hasło, oraz powtarza je w odpowiednich polach formularza. 4. Użytkownik wciska przycisk „Zmień”/”Ustaw hasło”. 5. Wysyłane zostaje na serwer zapytanie mające na celu zrealizowanie wybranej czynności. 6. Użytkownik zostaje poinformowany o wyniku operacji.
<b>Warunki końcowe</b>	Użytkownik otrzymuje z serwera wiadomość potwierdzającą wykonanie operacji.
<b>Sytuacje wyjątkowe</b>	1. Otrzymanie z serwera kodu błędu. 2. Brak dostępu do Internetu.
<b>Rozwiązywanie sytuacji wyjątkowych</b>	Wyświetlenie użytkownikowi komunikatu. W przypadku wygaśnięcia tokenu autoryzacyjnego następuje przekierowanie do aktywności logowania.

Tabela 3.12. Wyświetlanie historii treningów

<b>Nazwa przypadku użycia</b>	<b>Wyświetlanie historii treningów</b>
<b>Zdarzenie wywołujące</b>	Użytkownik przycisk „Historia treningów”
<b>Opis</b>	Obsługuje wyświetlanie historii treningów użytkownika.
<b>Aktorzy</b>	Użytkownik, Serwer
<b>Warunki początkowe</b>	Uprawnienia zalogowanego użytkownika, połączenie z Internetem
<b>Przebieg</b>	1. Wysyłane jest do serwera zapytanie mające na celu pobranie historii treningów użytkownika. 2. Po otrzymaniu odpowiedzi z serwera widok jest aktualizowany. 3. Użytkownik wybiera trening którego historie chce zobaczyć. 4. Wyświetlone zostają dane wybranego treningu.
<b>Warunki końcowe</b>	Użytkownik klika przycisk „back” na urządzeniu.
<b>Sytuacje wyjątkowe</b>	1. Otrzymanie z serwera kodu błędu. 2. Brak dostępu do Internetu.
<b>Rozwiązywanie sytuacji wyjątkowych</b>	Wyświetlenie użytkownikowi komunikatu. W przypadku wygaśnięcia tokenu autoryzacyjnego następuje przekierowanie do aktywności logowania.

Tabela 3.13. Wybór rodzaju treningu

<b>Nazwa przypadku użycia</b>	<b>Wybór rodzaju treningu</b>
<b>Zdarzenie wywołujące</b>	Użytkownik klika w opcje menu „Trening”
<b>Opis</b>	Obsługuje wybór rodzaju treningu
<b>Aktorzy</b>	Użytkownik
<b>Warunki początkowe</b>	Uprawnienia zalogowanego użytkownika (W przypadku niezalogowanego użytkownika dostępny jest tylko jeden rodzaj treningu)



Nazwa przypadku użycia	Wybór rodzaju treningu
Przebieg	<ol style="list-style-type: none"> <li>1. Użytkownik klika przycisk trening</li> <li>2. Wyświetlone zostają możliwe treningi.</li> <li>3. Użytkownik klika wybrany trening.</li> <li>4. Opcjonalnie, przy wyborze treningu po trasie, użytkownik zostaje przekierowany do wyboru trasy.</li> </ol>
Warunki końcowe	Rozpoczęcie treningu.

Tabela 3.14. Rozpoczęcie treningu

Nazwa przypadku użycia	Rozpoczęcie treningu
Zdarzenie wywołujące	Wybranie rodzaju treningu i Kliknięcie przycisku rozpocznij trening
Opis	Obsługuje rozpoczęcie treningu
Aktorzy	Użytkownik, Serwer
Warunki początkowe	Uprawnienia zalogowanego użytkownika(Za wyjątkiem treningu niezalogowanego użytkownika), połączenie z Internetem(Za wyjątkiem treningu niezalogowanego użytkownika)
Przebieg	<ol style="list-style-type: none"> <li>1. Uruchomienie serwisu GPS.</li> <li>2. Po otrzymaniu startowej lokalizacji wysłanie informacji na serwer o utworzeniu sesji treningowej.</li> <li>3. Odblokowanie przycisku Rozpocznij bieg.</li> <li>4. Użytkownik wciska przycisk rozpocznij bieg.</li> <li>5. Rozpoczęcie pomiarów.</li> <li>6. (W przypadku niezalogowanego użytkownika komunikacja z serwerem jest pomijana)</li> </ol>
Warunki końcowe	Rozpoczęcie pomiarów i aktualizacja widoku
Sytuacje wyjątkowe	<ol style="list-style-type: none"> <li>1. Otrzymanie z serwera kodu błędu.</li> <li>2. Brak dostępu do Internetu.</li> <li>3. Niemożliwość ustalenia pozycji przez GPS.</li> </ol>
Rozwiązywanie sytuacji wyjątkowych	Wyświetlenie użytkownikowi komunikatu. W przypadku wygaśnięcia tokenu autoryzacyjnego następuje przekierowanie do aktywności logowania.

Tabela 3.15. Trening

Nazwa przypadku użycia	Trening
Zdarzenie wywołujące	Rozpoczęcie pomiarów
Opis	Obsługuje przebieg treningu
Aktorzy	Użytkownik, Serwer(Za wyjątkiem treningu niezalogowanego użytkownika)
Warunki początkowe	Uprawnienia zalogowanego użytkownika(Za wyjątkiem treningu niezalogowanego użytkownika), połączenie z Internetem(Za wyjątkiem treningu niezalogowanego użytkownika)
Przebieg	<ol style="list-style-type: none"> <li>1. Aktualizowanie widoku danych treningowych.</li> <li>2. Cykliczne wysyłanie aktualnych danych treningowych na serwer.</li> <li>3. Odbieranie i odczytywanie wiadomości od trenera.</li> <li>4. Opcjonalnie możliwe jest robienie zdjęć podczas treningu. Użytkownik wciska przycisk zrób zdjęcie.</li> <li>5. W przypadku niezalogowanego użytkownika komunikacja z serwerem jest pomijana.</li> </ol>
Warunki końcowe	Wciśnięcie przycisku „Zakończ trening”
Sytuacje wyjątkowe	<p>Otrzymanie z serwera kodu błędu.</p> <p>Brak dostępu do Internetu.</p> <p>Niemożliwość ustalenia pozycji przez GPS.</p>
Rozwiązywanie sytuacji wyjątkowych	Wyświetlenie użytkownikowi komunikatu..

Tabela 3.16. Wyświetlanie podsumowania treningu

Nazwa przypadku użycia	Wyświetlanie podsumowania treningu
Zdarzenie wywołujące	Zakończenie treningu
Opis	Obsługuje wyświetlanie użytkownikowi ekranu podsumowania treningu.
Aktorzy	Użytkownik, Serwer(Za wyjątkiem treningu niezalogowanego użytkownika)
Warunki początkowe	Uprawnienia zalogowanego użytkownika(Za wyjątkiem treningu niezalogowanego użytkownika), połączenie z Internetem(Za wyjątkiem treningu niezalogowanego użytkownika)
Przebieg	<ol style="list-style-type: none"> <li>1. Przekierowanie użytkownika do ekranu wyświetlającego podsumowanie treningu. Ekran ten zawiera dane treningu i miniaturę trasy przebytej przez użytkownika.</li> <li>2. Użytkownik widzi przyciski umożliwiające powrót to głównej aktywności oraz wysłanie danych treningowych lub trasy na serwer.</li> </ol>

Nazwa przypadku użycia	Wyświetlanie podsumowania treningu
Warunki końcowe	Wciśnięcie przycisku „Powrót do Menu”
Sytuacje wyjątkowe	Brak połączenia z Internetem.
Rozwiązywanie sytuacji wyjątkowych	Miniatura trasy nie zostanie wyświetlona.

### 3.3. Serwer(Paweł Mazurek)

Serwer ma na celu udostępnić szereg zapytań – tzw. API, wszystkim zainteresowanym podmiotom, czyli aplikacji mobilnej i webowej. W odpowiedzi na wykonane zapytania(*ang. request*) serwer odpowiada wiadomością(*ang. response*). Zapytania będą wysyłane przez protokół HTTP, więcej informacji na jego temat znajdują się w rozdziale 4.3. Serwer pozwala na komunikację dwóch komponentów opisywanego systemu (aplikacji mobilnej i webowej), zapisywanie danych dostarczanych przez nie, oraz zarządzanie tymi danymi. Zapisuje również dane pogodowe z zewnętrznego serwisu (forecast.io) w pamięci podręcznej (proces ten ma swój angielski odpowiednik – *caching*) i udostępnia te dane wewnątrz systemu.

#### 3.3.1. Komponenty

Serwer składa się z kilku współdziałających komponentów, służących do obsługi: wszystkich przychodzących zapytań, użytkowników (w tym ich uwierzytelniania), serwisu przechowującego dane pogodowe z zewnętrznego serwera, treningów 'na żywo', treningów ukończonych, tras, modułu pozwalającego na akceptacje lub odrzucenie zapytań z prośbą o dodanie zawodnika do trenera. Komponenty te reprezentowane są przez osobne pliki ze skryptami obsługującymi daną funkcjonalność.

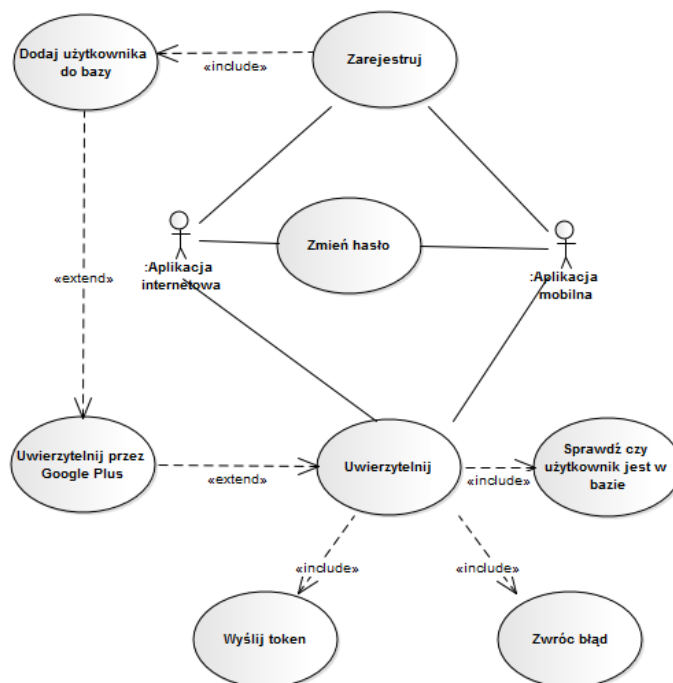
#### 3.3.2. Wymagania funkcjonalne

Tabela 3.17. Implementowane funkcje serwera

Redmine ID	Priorytet	Temat	Uwagi
90	Wysoki	API do podglądu treningu 'na żywo'	Możliwość rozpoczęcia treningu przez biegacza, zakończenie go, i udostępnienie tych danych trenerowi
89	Niski	Zapis tras trenera do swojego własnego konta	Trener ma mieć możliwość stworzenia trasy, oraz zapisania jej (z zachowaniem powiązania trasy z jej twórcą)
88	Normalny	System wysyłania wiadomości podczas treningu od trenera do zawodnika	Podczas gdy trening 'na żywo' jest uruchomiony, należy udostępnić trenerowi możliwość wysyłania wiadomości zawodnikowi
87	Normalny	Udostępnianie tras stworzonych przez trenera biegaczowi	Trasy stworzone przez trenera należy udostępnić biegaczowi
86	Wysoki	Możliwość dodawania zawodnika do trenera	Możliwość dwustronnego wysłania zapytania oraz jego odrzucenie / akceptacja
85	Niski	Zapisywanie zdjęć	Zdjęć trasy
84	Niski	Udostępnianie zdjęć	Zdjęć trasy
127	Pilny	Uwierzytelnianie użytkowników	
128	Niski	Udostępnienie danych	Po podaniu współrzędnych, serwer zwraca dla

Redmine ID	Priorytet	Temat	Uwagi
		pogodowych	nich pogodę na najbliższe 24 godziny
129	Normalny	Zapisywanie ukończonego treningu	

### 3.3.3. Przypadki użycia



3.3. Diagram przypadków użycia odpowiedzialnych za zarządzanie użytkownikami

Opisane zostały tylko najważniejsze z nich. Aby praca była czytelniejsza i bardziej przejrzysta, pominąłem opis najbardziej oczywistych.

Nazwa przypadku użycia	Zarejestruj
Zdarzenie wywołujące	Użytkownik wysyła odpowiednie zapytanie POST
Opis	Obsługuje rejestrację użytkowników. Serwer po otrzymaniu odpowiednich danych dodaje użytkownika do bazy danych
Aktorzy	Aplikacja mobilna i webowa
Warunki początkowe	Użytkownik wysyła zapytanie do serwera
Przebieg	<ol style="list-style-type: none"> <li>1. Użytkownik tworzy zapytanie</li> <li>2. Użytkownik zawiera dane nowego użytkownika w ciele zapytania</li> <li>3. Użytkownik wysyła zapytanie</li> <li>4. Serwer weryfikuje poprawność danych (sprawdza czy hasło nie jest puste, i czy nazwa użytkownika jest adresem e-mail)</li> <li>5. Serwer dodaje użytkownika do bazy danych</li> <li>6. Serwer wysyła wiadomość potwierdzającą dodanie użytkownika do bazy danych</li> </ol>
Warunki końcowe	Użytkownik otrzymuje od serwera wiadomość potwierdzającą dodanie go do bazy, wraz z kodem HTTP 200

Nazwa przypadku użycia	Uwierzytelnij
Zdarzenie	Użytkownik wysyła odpowiednie zapytanie POST

<b>wywołujące</b>	
<b>Opis</b>	Obsługuje uwierzytelnianie użytkowników, poprzez sprawdzenie czy istnieje w bazie danych, oraz czy hasło jest poprawne. Serwer odpowiada tokenem uwierzytelniającym
<b>Aktorzy</b>	Aplikacja mobilna i webowa
<b>Warunki początkowe</b>	Użytkownik posiada konto w bazie danych, oraz dostarczy poprawne dane
<b>Przebieg</b>	<ol style="list-style-type: none"> <li>1. Użytkownik tworzy zapytanie</li> <li>2. Użytkownik zawiera dane użytkownika w ciele zapytania</li> <li>3. Użytkownik wysyła zapytanie</li> <li>4. Serwer weryfikuje poprawność danych</li> <li>5. Serwer wysyła token uwierzytelniający</li> </ol>
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera token uwierzytelniający, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Zmień hasło</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie POST
<b>Opis</b>	Obsługuje usługę zmiany hasła przez użytkownika
<b>Aktorzy</b>	Aplikacja mobilna i webowa
<b>Warunki początkowe</b>	<p>Użytkownik wysyła zapytanie do serwera.</p> <p>Użytkownik jest uwierzytelniony.</p>
<b>Przebieg</b>	<ol style="list-style-type: none"> <li>1. Użytkownik tworzy zapytanie</li> <li>2. Użytkownik zawiera nowe hasło w ciele zapytania</li> <li>3. Użytkownik wysyła zapytanie</li> <li>4. Serwer weryfikuje poprawność danych (sprawdza czy hasło nie jest puste)</li> <li>5. Serwer zmienia hasło użytkownika w bazie danych</li> <li>6. Serwer wysyła wiadomość potwierdzającą zmianę hasła</li> </ol>
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera wiadomość potwierdzającą zmianę hasła, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Uwierzytelnij przez Google Plus</b>
<b>Zdarzenie wywołujące</b>	Użytkownik loguje się przez Google Plus
<b>Opis</b>	Obsługuje uwierzytelnianie użytkownika, jeżeli zalogował się przez usługę Google Plus
<b>Aktorzy</b>	Aplikacja mobilna i webowa
<b>Warunki początkowe</b>	<p>Użytkownik zalogował się przez Google Plus, serwery Google zwróciły mu token.</p> <p>Użytkownik wysyła zapytanie do serwera.</p>
<b>Przebieg</b>	<ol style="list-style-type: none"> <li>1. Użytkownik tworzy zapytanie informujące serwer że loguję się przez Google Plus</li> <li>2. Użytkownik zawiera swoją nazwę użytkownika w ciele zapytania,</li> <li>3. Użytkownik wysyła zapytanie</li> <li>4. Serwer weryfikuje poprawność danych (sprawdza czy użytkownik istnieje)</li> <li>5. Serwer zwraca użytkownikowi token uwierzytelniający, lub jeżeli nie znalazł go w bazie, to tworzy w niej nowy wpis</li> </ol>
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera token uwierzytelniający, wraz z kodem HTTP 200



x. Diagram pozostałych przypadków użycia dla serwera

Uwaga – przypadek użycia „Uwierzytelnij” jest tożsamy z przypadkiem użycia o tej samej nazwie z poprzedniego diagramu przedstawionego na rysunku [NUMER RYSUNKU]. Diagramy zostały rozdzielone dla większej czytelności.

#### Opisy przypadków użycia

<b>Nazwa przypadku użycia</b>	<b>Pobierz dane pogodowe</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysła odpowiednie zapytanie GET
<b>Opis</b>	Serwer udostępnia dane pogodowe, przechowywane we własnej pamięci podręcznej.
<b>Aktorzy</b>	Aplikacja mobilna i webowa
<b>Warunki początkowe</b>	Użytkownik wysła zapytanie do serwera
<b>Przebieg</b>	<ol style="list-style-type: none"> <li>1. Użytkownik tworzy zapytanie, w którym zawiera szerokość i długość geograficzną punktu dla którego pogodę chce sprawdzić</li> <li>2. Serwer przeszukuje pamięć podręczną w poszukiwaniu odpowiednich</li> </ol>

	danych 3. Serwer odpowiada użytkownikami danymi pogodowymi z formacie JSON
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera dane pogodowe, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Pobierz listę zapytań o dodanie zawodnika do trenera</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie GET
<b>Opis</b>	Serwer odpowiada użytkownikowi listą wszystkich próśb (wysłanych przez trenerów) o dodanie zawodnika jako własnego biegacza
<b>Aktorzy</b>	Aplikacja mobilna (zawodnik)
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera Użytkownik jest uwierzytelniony
<b>Przebieg</b>	1. Użytkownik wysyła do serwera prośbę o zwrócenie listy zapytań wysłanych do jego własnego konta 2. Serwer odpowiada listą zapytań, która w przypadku braku zapytań może być pusta
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera listę zapytań, w której znajdują się numery identyfikacyjne zapytań, oraz nazwy użytkownika trenerów którzy wysłali te zapytania, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Pobierz listę zawodników</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie GET
<b>Opis</b>	Serwer odpowiada użytkownikowi (trenerowi) listą użytkowników obecnie przypisanych do jego konta (tych którzy potwierdzili zapytanie o dodanie zawodnika)
<b>Aktorzy</b>	Aplikacja webowa (trener)
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera Użytkownik jest uwierzytelniony
<b>Przebieg</b>	1. Trener wysyła do serwera prośbę o zwrócenie listy zawodników przypisanych do jego konta 2. Serwer odpowiada listą zawodników, w przypadku braku lista jest pusta
<b>Warunki końcowe</b>	Użytkownik otrzymuje od listę zawodników, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Pobierz username trenera</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie GET
<b>Opis</b>	Użytkownik pobiera nazwę użytkownika trenera który jest obecnie przypisany do jego konta
<b>Aktorzy</b>	Aplikacja mobilna (biegacz)
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera Użytkownik jest uwierzytelniony
<b>Przebieg</b>	1. Zawodnik wysyła do serwera zapytanie z prośbą o udostępnienie nazwy trenera obecnie dodanego do jego konta

	2. Serwer odpowiada nazwą trenera, lub informuje, że brak przypisanego trenera do konta
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera nazwę przypisanego trenera, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Zapisz trasę</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie POST
<b>Opis</b>	Serwer odbiera trasę wysłaną przez użytkownika i zapisuje ją do bazy danych
<b>Aktorzy</b>	Aplikacja mobilna i webowa
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera, w którego ciele umieszcza listę punktów trasy oraz dodatkowe jej parametry Użytkownik jest uwierzytelniony
<b>Przebieg</b>	1. Użytkownik wysyła zapytanie z informacją o całej trasie, czyli jej punktami, tytułem, opisem, czy chce by była widoczna dla wszystkich 2. Serwer waliduje poprawność wysłanych danych 3. Serwer zapisuje trasę do bazy danych
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera potwierdzenie zapisania trasy, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Zapisz trening</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie POST
<b>Opis</b>	Serwer odbiera trening wysłany przez użytkownika i zapisuje go do bazy danych
<b>Aktorzy</b>	Aplikacja mobilna (biegacz)
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera Użytkownik jest uwierzytelniony
<b>Przebieg</b>	1. Użytkownik wysyła zapytanie z informacją o całym wykonanym treningu, czyli jego tempie, długości, czasie trwania, liczbie spalonych kalorii, oraz trasą którą przebył 2. Serwer waliduje poprawność wysłanych danych 3. Serwer zapisuje trening do bazy danych
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera potwierdzenie zapisania treningu, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Rozpocznij trening</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie POST
<b>Opis</b>	Użytkownik rozpoczyna trening na żywo, czyli taki który będzie widoczny dla trenera
<b>Aktorzy</b>	Aplikacja mobilna (biegacz)
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera Użytkownik jest uwierzytelniony

<b>Przebieg</b>	<ol style="list-style-type: none"> <li>1. Zawodnik wysyła do serwera zapytanie z prośbą o rozpoczęcie treningu na żywo, zawiera w nim pierwszy punkt w którym się aktualnie znajduje</li> <li>2. Serwer sprawdza czy dany użytkownik już nie rozpoczął treningu, jeżeli nie, to dodaje trening do pamięci serwera (nie do bazy danych) i zwraca informacje potwierdzającą</li> </ol>
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera potwierdzenie rozpoczęcia treningu, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Zaktualizuj trening</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie POST
<b>Opis</b>	Użytkownik wysyła serwerowi swoją aktualną pozycję, która zostanie dopisana do rozpoczętego treningu przez serwer. Zapytanie to jest wysyłane przez użytkownika cyklicznie.
<b>Aktorzy</b>	Aplikacja mobilna (biegacz)
<b>Warunki początkowe</b>	<p>Użytkownik wysyła zapytanie do serwera</p> <p>Użytkownik jest uwierzytelniony</p>
<b>Przebieg</b>	<ol style="list-style-type: none"> <li>1. Zawodnik wysyła do serwera zapytanie w którym zawiera współrzędne w którym aktualnie się znajduje</li> <li>2. Serwer sprawdza czy dany użytkownik rozpoczął trening, jeżeli tak, to aktualizuje trening i zwraca informacje potwierdzającą zaktualizowanie treningu</li> <li>3. Serwer zwraca listę wiadomości wysłanych przez trenera do zawodnika w trakcie treningu</li> </ol>
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera potwierdzenie zaktualizowania treningu, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Zakończ trening</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie POST
<b>Opis</b>	Użytkownik kończy wcześniej rozpoczęty trening na żywo
<b>Aktorzy</b>	Aplikacja mobilna (biegacz)
<b>Warunki początkowe</b>	<p>Użytkownik wysyła zapytanie do serwera</p> <p>Użytkownik jest uwierzytelniony</p>
<b>Przebieg</b>	<ol style="list-style-type: none"> <li>1. Zawodnik wysyła do serwera zapytanie z prośbą o zakończenie treningu na żywo</li> <li>2. Serwer sprawdza czy dany użytkownik rozpoczął treningu, jeżeli tak, to usuwa go z pamięci i zwraca informację potwierdzającą</li> </ol>
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera potwierdzenie zakończenia treningu, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Pobierz listę aktualnie prowadzonych treningów</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie GET
<b>Opis</b>	Trener pobiera listę jego zawodników, którzy są aktualnie podczas treningu na żywo



<b>Aktorzy</b>	Aplikacja webowa (trener)
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera Użytkownik jest uwierzytelniony
<b>Przebieg</b>	1. Trener wysyła do serwera zapytanie o udostępnienie listy zawodników którzy aktualnie trenują 2. Serwer odpowiada listą obecnie trenujących zawodników przypisanych do konta trenera, może być pusta w przypadku braku aktywnych treningów.
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera listę rozpoczętych treningów przez własnych zawodników, wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Zapisz zdjęcie</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie POST
<b>Opis</b>	Zawodnik wysyła do serwera zapytanie, w którego ciele umieszcza kod base64 zdjęcia. Serwer odczytuje go, zapisuje we własnym systemie plików, oraz umieszcza adres URL tego zdjęcia w bazie danych, w tabeli tras, ponieważ zdjęcia są powiązane z trasami.
<b>Aktorzy</b>	Aplikacja mobilna (biegacz)
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera Użytkownik jest uwierzytelniony
<b>Przebieg</b>	1. Użytkownik wysyła w zapytaniu kod base64 obrazka, oraz z informację z jaką trasą chce powiązać obrazek 2. Serwer zapisuje obrazek w systemie plików, oraz adres URL w bazie danych.
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera potwierdzenie zapisania zdjęcia wraz z kodem HTTP 200

<b>Nazwa przypadku użycia</b>	<b>Pobierz listę tras publicznych</b>
<b>Zdarzenie wywołujące</b>	Użytkownik wysyła odpowiednie zapytanie GET
<b>Opis</b>	Zawodnik wysyła do serwera zapytanie o udostępnienie listy tras publicznych
<b>Aktorzy</b>	Aplikacja mobilna i webowa
<b>Warunki początkowe</b>	Użytkownik wysyła zapytanie do serwera Użytkownik jest uwierzytelniony
<b>Przebieg</b>	1. Użytkownik wysyła odpowiednie zapytanie 2. Serwer zapisuje odpowiada użytkownikowi listą tras, łącznie z opcjonalnymi adresami URL zdjęć wykonanych na danej trasie
<b>Warunki końcowe</b>	Użytkownik otrzymuje od serwera listę tras publicznych, łącznie z ich listą punktów, adresami URL zdjęć i kodem HTTP 200

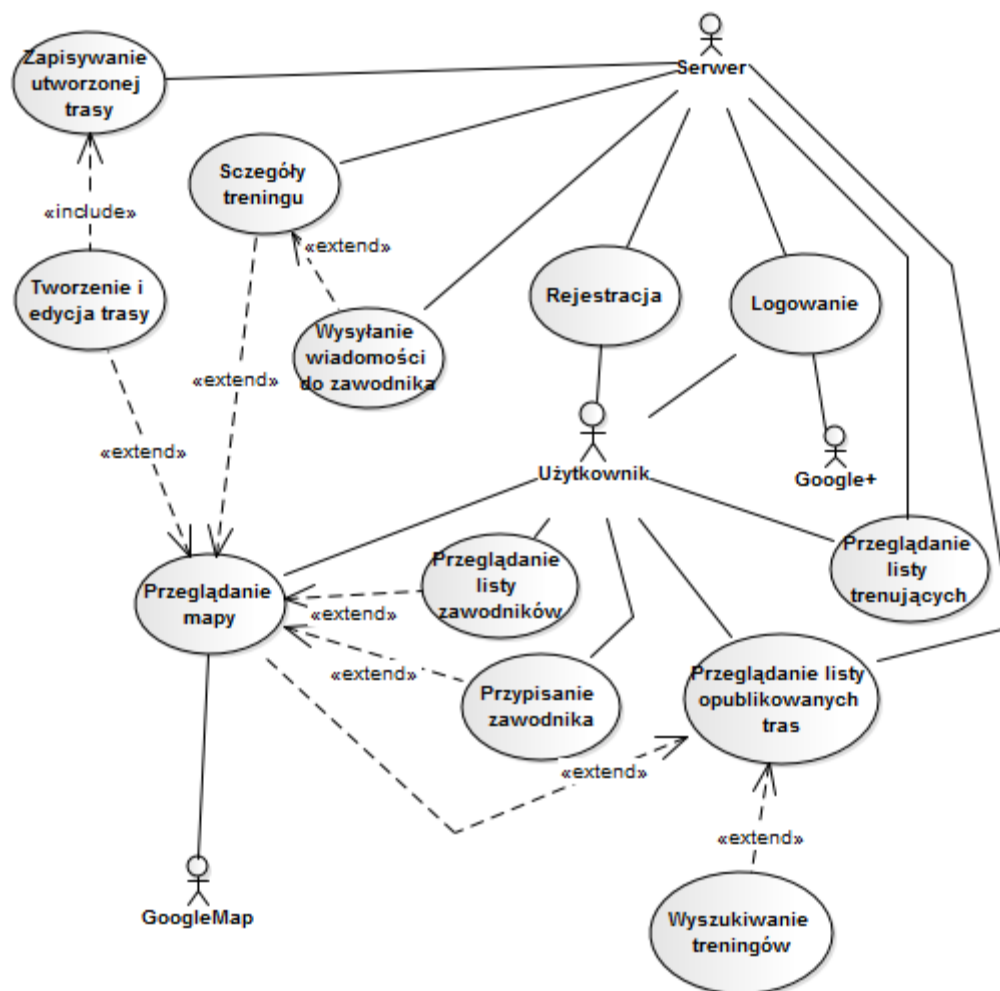
### 3.4. *Aplikacja trenera*(Marcin Olszewski)

#### 3.4.1. *Wymagania funkcjonalne*

ID z systemu	Priorytet	Temat	Uwagi
--------------	-----------	-------	-------

redmine			
70	Niski	Przeglądanie galerii zdjęć z trasy	
69	Niski	Wykreślanie profilu wysokościowego trasy	
68	Normalny	Listowanie zawodników, którzy obecnie trenują	
67	Normalny	Dodawanie sobie zawodnika	
66	Normalny	Zlecanie treningu przez trenera jego zawodnikowi	
65	Normalny	Listowanie tras treningowych przygotowanych przez trenera	
64	Normalny	Zapis tras treningowych	
63	Wysoki	Kontrola uprawnień - opcje trenera	
62	Normalny	Wysyłanie wiadomości do zawodnika	
10	Normalny	Podgląd trasy zawodnika na żywo	
9	Normalny	Listowanie tras opublikowanych dla wszystkich	

#### 3.4.2. Przypadki użycia



## 4. ARCHITEKTURA SYSTEMU ORAZ PRZEGLĄD TECHNOLOGII

### 4.1. Ogólna architektura całego systemu.

### 4.2. Aplikacja mobilna

<https://github.com/RunandPL/AndroidApp>

#### Google Plus

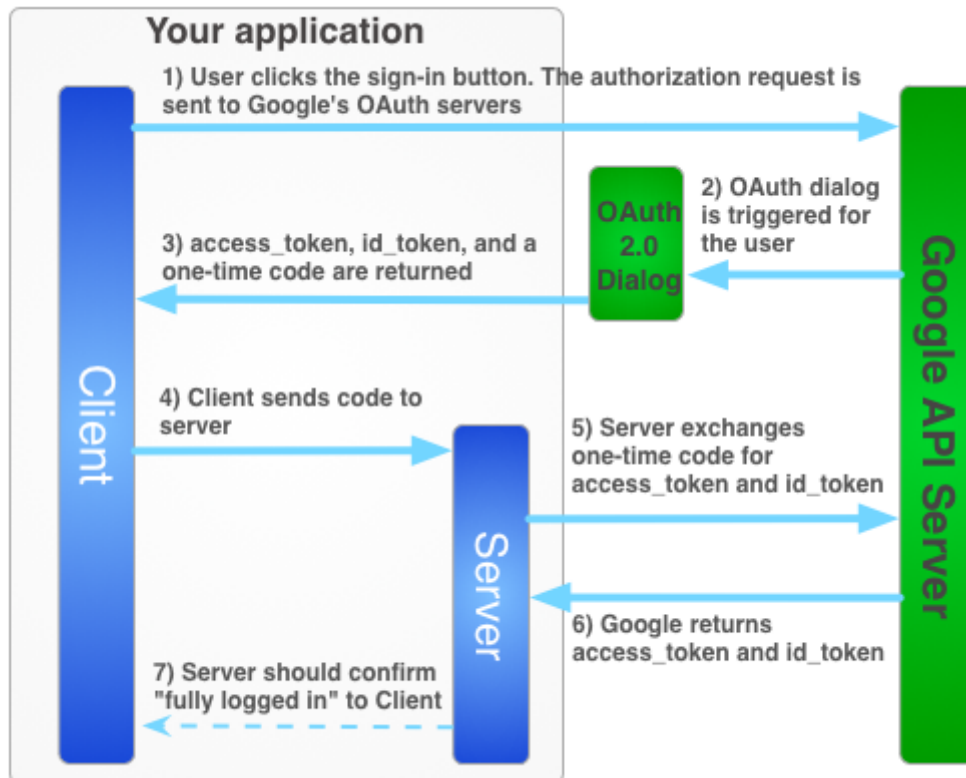
Zgodnie z założeniami przedstawionymi na etapie wizji systemu i analizy funkcjonalnej wprowadziliśmy do naszego systemu możliwość uwierzytelniania użytkowników za pomocą istniejącego konta Google. Udostępnienie takiego systemu logowania ma na celu znaczną poprawę wygody użytkownika aplikacji. Taki rodzaj uwierzytelniania pozwala użytkownikom na bezpieczne zalogowanie się do naszego systemu za pomocą danych już wprowadzonych do telefonu podczas personalizacji Systemu Android, lub za pomocą loginu i hasła wykorzystywanego na co dzień do korzystania chociażby z poczty elektronicznej Gmail czy serwisu YouTube. Użytkownik chcący przeprowadzić trening ma możliwość zalogowania się na swoje konto jednym kliknięciem w aplikacji,

bez konieczności niewygodnego podawania loginu i hasła. Również aplikacja dla trenerów umożliwia taki sposób logowania.

Dodatkową zaletą używania Google+ do logowania i uwierzytelniania jest wprowadzony w 2011 roku przez firmę Google system umożliwiający dodanie opcjonalnego, dwustopniowego mechanizmu weryfikacji, który może znacząco zmniejszyć ryzyko utraty kontroli nad własnym kontem. Potwierdzenie logowania Google przypomina potwierdzenie wykonania przelewu kodem jednorazowym przysyłanym przez SMS. Właściciel konta może zdecydować czy chce to robić przy każdym logowaniu czy raz na 30 dni.

Logowanie i uwierzytelnianie użytkowników po stronie serwera Google odbywa się w zapewnionej przez Google bibliotece wykorzystującej i zarządzającej protokołem OAuth 2.0. Google wspiera popularne scenariusze OAuth 2.0 dla aplikacji webowych, serwerowych, preinstalowanych na urządzeniu, oraz działających po stronie klienta. OAuth jest otwartym standardem autoryzującym. Umożliwia dzielenie się przez użytkowników zasobami bez konieczności zagłębiania się w obsługę ich poświadczeń, dostarczając zazwyczaj nazwę użytkownika i token autoryzujący. Token daje możliwość dostępu do określonej strony oraz zasobów na określony czas. Pozwala to udzielić dostęp do informacji przechowywanych w serwisie bez konieczności współdzielenia praw dostępu i bez udzielania pełnego dostępu do nich. Dzięki temu nasz system może uzyskać podstawowe informacje o koncie Google użytkownika. OAuth jest usługą uzupełniającą, lecz odrębną od OpenID, które rozwiązuje problem dystrybucji składników tożsamości użytkownika serwisu takich jak imię nazwisko, adres email pomiędzy wieloma serwisami webowymi. Podczas logowania do pożądanego serwisu użytkownik podaje w formie adresu URL, a od wersji 2.0 protokołu OpenID identyfikator użytkownika może mieć formę XRI (eXtensible Resource Identifier) swój identyfikator OpenID, który jest jednocześnie adresem serwera OpenID i wskazaniem konkretnego użytkownika. Serwis przekierowuje użytkownika na stronę serwera z żądaniem określonych danych, które udostępnia przez przekierowanie ich do serwisu który ich zażądał. Proces ten wymaga zwykle ścisłej interakcji z użytkownikiem, aby udostępnione zostały tylko te dane na których udostępnienie użytkownik się godzi i tylko tym serwisom które użytkownik akceptuje. Dzięki metodom zawartym w Google PLAY API developer nie musi oprogramowywać całego wyżej wymienionego procesu.

Ogólny proces uwierzytelniania użytkowników za pomocą konta Google[rysunek xxx [https://developers.google.com/+/images/server\\_side\\_code\\_flow.png](https://developers.google.com/+/images/server_side_code_flow.png)] (w przypadku wersji na aplikacje mobilne i webowe występują niewielkie różnice) przedstawia się następująco:



1. Użytkownik klika przycisk zaloguj się. Żądanie autoryzacji wysyłane jest do serwerów OAuth Google.
2. Uruchamiany jest dialog OAuth dla użytkownika.
3. Tokeny: dostępu, id i jednorazowy token uwierzytelniania przekazywane są klientowi.
4. Klient wysyła dane na serwer aplikacji wraz z kodem jednorazowego uwierzytelniania.
5. Serwer potwierdza tożsamość klienta za pomocą kodu jednorazowego logowania.
6. Serwer potwierdza zalogowanie użytkownika do systemu.

### Biblioteka Knife (Mateusz Pakulski)

W ogólności strukturę projektu androidowego dzielimy na dwa obszary, a mianowicie obszar interfejsu oraz logikę, reagującą na komendy wydawane podczas interakcji użytkownika z interfejsem i zmieniającą interfejs zależnie od wewnętrznego stanu aplikacji. Niemal każda aplikacja pisana dla systemu operacyjnego Android posiada interaktywny interfejs przeważnie w formie okna z kontrolkami takimi jak przyciski pola tekstowe czy checkboxy.



Praca nad standardowym projektem androidowym polega na stworzeniu interfejsu aktywności w formie plików XML, a następnie podpięciu logiki do każdego elementu, który tego wymaga. Logika bardzo prostej aplikacji zawierającej tylko jedną aktywność z dwoma kontrolkami przyciskiem i polem tekstowym, w którym po wciśnięciu przycisku zmienia się tekst wygląda następująco:

1. `public class MainActivity extends ActionBarActivity {`
- 2.

```

3.     TextView text;
4.     Button button;
5.
6.     @Override
7.     protected void onCreate(Bundle savedInstanceState) {
8.         super.onCreate(savedInstanceState);
9.         setContentView(R.layout.activity_main);
10.        text = (TextView) this.findViewById(R.id.text);
11.        button = (Button) this.findViewById(R.id.button);
12.        text.setText("Hello World!");
13.
14.        button.setOnClickListener(new View.OnClickListener() {
15.            @Override
16.            public void onClick(View view) {
17.                text.setText("Goodbye, World!");
18.            }
19.        });
20.    }
21.
22.    }

```

Na początku w metodzie *onCreate()* wyszukiwana są wszystkie elementy interfejsu użytkownika na interakcje z którymi będziemy reagować, lub których będą zmieniane po takiej interakcji [linie 10 i 11]. Dopiero po wyszukaniu elementów możliwe jest odwołanie się do nich z kodu aplikacji i wpływanie na ich wygląd bądź edycja wartości [linia 17]. Standardowy sposób wyszukiwania elementów posiada jednak wadę znacząco zmniejszającą wygodę pisania kodu. Jest to konieczność rzutowania każdego wyszukiwanego elementu na odpowiedni typ mimo iż obiekt utworzony do przechowywania reprezentacji tego elementu posiada jawnie zadeklarowany typ [linie 10 i 11]. Przedstawiony powyżej listing aplikacji nie jest bardzo skomplikowany, jednak już tak prosty program zawiera nietrywialne wywołanie jakim jest tworzenie obiektu w parametrze innej metody [linie 14-19].

Standardowe aktywności składają się zazwyczaj z kilkunastu elementów komponentów które trzeba zainicjalizować, bądź reagować na ich zmiany. Wszystkie te komponenty muszą zostać znalezione w hierarchii layoutu, ich handlersy muszą zostać zarejestrowane, a logika obsługująca ich zdarzenia musi zostać napisana. Prowadzi to do niepotrzebnego skomplikowania kodu.

Na szczęście istnieje sporo rozwiązań tego problemu. Jednym z nich jest biblioteka ButterKnife stworzona przez Jake'a Wharton'a, wykorzystaną w trakcie tworzenia projektu. Ułatwia ona wykorzystanie adnotacji do wygodnego wstrzykiwania widoków androidowych. Ułatwia ona dwie bardzo często wykonywane na etapie tworzenia aplikacji czynności – wyszukiwanie komponentów i podpinanie kliknięć jednocześnie sprawiając, że kod staje się prostszy do późniejszej analizy. Poniżej listing przedstawia tę samą aplikację napisaną z wykorzystaniem możliwości udostępnianych przez bibliotekę ButterKnife.

```

1.     public class MainActivity extends ActionBarActivity {
2.         @InjectView(R.id.text) TextView text;
3.
4.         @OnClick(R.id.button) void button() {

```

```

5.     text.setText("Goodbye, World!");
6.     }
7.
8.     @Override
9.     protected void onCreate(Bundle savedInstanceState) {
10.        super.onCreate(savedInstanceState);
11.        setContentView(R.layout.activity_main);
12.
13.        ButterKnife.inject(this); // wstrzyknięcie zależności
14.
15.        text.setText("Hello World!");
16.    }

```

Im bardziej skomplikowana hierarchia, tym większa jest oszczędność czasu i ilości nadmiarowego kodu. Biblioteka ButterKnife umożliwia również podpinanie innych zdarzeń takich jak chociażby `@OnLongClick` czy `@OnItemSelected`.

ButterKnife generuje i używa klasy dynamicznie co w niektórych przypadkach może powodować niechciane opóźnienia w działaniu aplikacji. W przypadku klasycznych aplikacji opóźnienia te są jednak niezauważalnie małe i w żaden sposób nie wpływają na płynność działania aplikacji. Dynamiczna generacja i użycie klas niesie za sobą również konieczność poprawnego skonfigurowania statycznych narzędzi analizy kodu takich jak ProGuard. W przypadku braku konfiguracji niektóre metody dołączone za pomocą adnotacji mogą zostać wykryte jako nieużywane i usunięte na etapie kompilacji.

#### 4.3. Serwer (Paweł Mazurek)

<https://github.com/RunandPL/Serwer>

##### REST

Na etapie projektowania aplikacji, zdecydowaliśmy się na to by serwer został napisany z wykorzystaniem wzorca architektury oprogramowania REST. Skrót ten oznacza Representational State Transfer. Jego głównymi cechami jest jednorodny interfejs oraz bezstanowa komunikacja z klientem. W przeciwieństwie do klasycznego sposobu tworzenia aplikacji serwerowych, w REST parametry wywołania danej usługi, często są umieszczane w ścieżce adresu URL, lecz nie jest to wymagane. REST opiera się na zasobach i zarządzaniu nimi odpowiednimi zapytaniami HTTP. Wzorzec ten został zaproponowany w roku 2000 przez Roya Thomasa Fieldinga w postaci jego pracy doktorskiej, pt. *„Architectural Styles and the Design of Network-based Software Architectures”* broniącej na uniwersytecie kalifornijskim. Środowiskiem na który zdecydowaliśmy się do implementacji serwera, jest serwer Node.js.

**Node.js** (<http://nodejs.org/>)



Jest to platforma zbudowana na silniku Javascript Chrome. Cały kod serwera napisany jest więc w Javascript, języku który pierwotnie powstał do zupełnie innych celów. Stworzyła go firma Netscape w 1995 roku, a miał służyć do ożywienia statycznych stron Internetowych, poprzez dodanie do nich interakcji, takich jak: zdarzeń wywoływanych kliknięciami czy przewijaniem strony, różnego rodzaju animacji. Język ten jest językiem interpretowanym, co oznacza, że użytkownik aplikacji JavaScript otrzymuje kod źródłowy programu, który jest interpretowany na żywo, przez różnego rodzaju silniki stworzone do tego celu.

Ryan Dahl, twórca Node.js pracujący w tamtym czasie dla firmy Joyent, miał inny pomysł na wykorzystanie potencjału języka Javascript. W roku 2009 miał na celu zaimplementowanie technologii push dla stron Internetowych (polega na automatycznym przesyłaniu danych do odbiorców). Po wypróbowaniu kilku języków, zdecydował się na Javascript, co dało podwaliny pod Node.js, technologii asynchronicznej, sterowanej zdarzeniami, opartej o nie blokujące operacje wejścia / wyjścia. Jest ona przeznaczona do wytwarzania wysoce skalowalnych aplikacji serwerowych, co potwierdzają testy przy obciążaniu przykładowej aplikacji Node.js nawet milionem użytkowników, wysyłających zapytania do serwera.

Node.js jest sterowany zdarzeniami, co oznacza, że gdy zestawimy go z klasycznym modelem aplikacji serwerowych (gdzie sieciowość oparta jest na wątkach), okazuje się, że w modelu klasycznym występują tzw. Dead-locki, z którymi walką obciążony jest programista. W Node.js prawie żadna funkcja nie wykonuje bezpośrednich operacji wejścia wyjścia, dzięki czemu jest on środowiskiem nie blokującym, w którym programowanie jest dużo łatwiejsze.



Node.js jest bardzo modularny, pozwala na wykorzystywanie w nim osobnych modułów, często stworzonych przez entuzjastów Node.js. Moduły te są ładowane poprzez manager pakietów, NPM - *Node Packaged Modules*, który opiszę w osobnym punkcie.

Na stronach Internetowych, skrypty w Javascript są zazwyczaj ładowane poprzez użycie tagu HTML `<script>`. Z racji tego, że w Node.js HTML nie jest wykorzystywany do niczego, moduły ładowane poprzez AMD(*ang. Asynchronous Module Definition*). Jest to asynchroniczny sposób ładowania modułów, który dba o nie występowanie konfliktów podczas definiowania zależności pomiędzy modułami.

**NPM** (<https://www.npmjs.org/>)

Jest to manager pakietów stworzony dla Node.js. Umieszcza on moduły w jednym miejscu, co pozwala na łatwą lokalizację oraz zarządzanie nimi. Zarządza on również inteligentnie konfliktami występującymi pomiędzy różnymi zależnościami, z których korzystają moduły. Twórcy zapewniają, że jest wysoce konfigurowalny, aby pokryć szeroką gamę przypadków użycia. Jest używany do publikowania, odkrywania, instalowania i tworzenia aplikacji Node.js. Tworząc projekt w Node.js tworzy się plik `package.json`, w którym znajdują się informacje, z jakich modułów korzysta aplikacja. Tak może wyglądać przykładowy plik:



```

1. {
2.   "name": "RunAnd-serwer",
3.   "version": "0.0.1",
4.   "dependencies": {
5.     "body-parser": "1.9.0",
6.     "q": "1.0.1"
7.   }
8. }

```

W linii 6 przedstawiona jest informacja, że dana aplikacja korzysta z biblioteki Q.js, w wersji 1.0.1. W przypadku gdy chce się zawsze używać najnowszej wersji dostępnej biblioteki, w polu numeru wersji, można wpisać „latest”. Może to w przyszłości spowodować wiele nieprzewidzianych problemów. Na przykład, wersja biblioteki się zmieniła, na taką w której stare API nie jest wspierane, aplikacja pobrała najnowszą jej wersję, i aplikacja przestaje działać.

Aby zaktualizować wszystkie zależności aplikacji, jeżeli w folderze jest zdefiniowany plik package.json, będąc w folderze projektu, wystarczy wpisać komendę `npm update`.

#### Q.js (<https://github.com/kriszowal/q>)

To biblioteka do tworzenia asynchronicznych obietnic w języku Javascript. Obietnice (*ang. Promises*) są wzorcem do programowania asynchronicznych programów. Polegają na tym, że w funkcji wykonującej jakieś asynchroniczne zadanie (np. odczyt z bazy danych), nie czekamy na odpowiedź z bazy danych przed zwróceniem rezultatu, przez co blokujemy wykonanie innych funkcji, tylko od razu zwracamy użytkownikowi obietnicę, że w przyszłości ona się wykona, (lub w wypadku błędu bazy danych zwróci błąd po pewnym czasie). Przykładowy kod wykorzystujący możliwości biblioteki Q.js:

```

1. function readDatabase() {
2.   var d = Q.defer();
3.   exampleReadFromDatabaseFunction().then(function() {
4.     d.resolve();
5.   },
6.   function(err) {
7.     d.reject();
8.   });
9.   return d.promise;
10. }

```

W linii 2, do zmiennej d, przypisywany jest obiekt „opóźniony” z Q.js. Posiada on między innymi pole „promise”, który jest obietnicą zwracaną użytkownikowi, że funkcja w przyszłości zwróci rezultat. W linii 3 wykonywana jest przykładowa asynchroniczna funkcja, zwracająca rezultat po pewnym czasie. Wykonanie funkcji resolve(), na obiekcie d, oznacza że funkcja asynchroniczna zakończyła działanie z powodzeniem, dzięki czemu użytkownik funkcji readDatabase(), otrzyma potwierdzenie wykonania metody. Analogicznie, w linii 7, np. w wyniku awarii bazy danych, zwrócony został błąd, i użytkownik funkcji głównej, tj. readDatabase() zostanie o tym

powiadomiony. Dodatkowo, w linii 3, funkcja `exampleReadFromDatabaseFunction()` również zwraca obietnicę. Aby przypisać akcję która zostanie wykonana po spełnieniu obietnicy, podaje się ją jako parametr funkcji `then()`. Akcja która wykona się w przypadku błędu, to drugi parametr tej funkcji (linia 6).

## PostgreSQL

Jest to relacyjna baza danych wykorzystywana przez nasz serwer, często również nazywany jako Postgres. Jest jedną z najbardziej rozpoznawalnych baz danych open source. Została wydana na licencji PostgreSQL License, bardzo podobnej do licencji BSD, czy MIT. Jest wytwarzana od ponad 15 lat. Posiada swoje implementacje na wszystkich najważniejszych systemach operacyjnych, tj. Linux, UNIX (Mac OS X, Solaris, Tru64), i Windows. Zawiera prawie wszystkie typy danych zdefiniowanych w SQL:2008, takie jak: INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, czy TIMESTAMP. Limity rozmiaru danych są bardzo wysokie, pokazane w tabeli nr [NR].

Limit	Wartość
Maksymalny rozmiar bazy danych	nielimitowany
Maksymalny rozmiar tabeli	32 TB
Maksymalny rozmiar wiersza	1.6 TB
Maksymalny rozmiar pola	1GB
Maksymalna liczba wierszy w tabeli	nielimitowana
Maksymalna liczba kolumn w tabeli	250 – 1600 (w zależności od typu kolumny)
Maksymalna liczba indeksów w tabeli	nielimitowana

Tabela [NR] Limity bazy danych PostgreSQL

(źródło: <http://www.postgresql.org/about/>)

## Bookshelf.js (<http://bookshelfjs.org/>)

To ORM(*ang. Object-Relational Mapping*) Javascriptowy wykorzystywany w implementacji serwera. ORM to sposób w jaki tabele bazy danych, są odwzorowywane w językach programowania na obiekty. Bookshelf.js jest zbudowany na podstawie kreatora zapytań Knex.js (który pozwala na ominięcie bezpośredniego tworzenia zapytań w czystym SQL). Poniższy przykład kodu źródłowego, pokazuje jak tworzy się w nim model na podstawie tabeli bazy danych, i dodaje się do niego informacje.

```
1. var Customer = bookshelf.Model.extend({  
2.   tableName: 'customers'  
});
```

```

3.
4. Customer.forge({item: 'value'}).save().then(function() {
5.     // ...
7. });
8.

```

W linii 1 do obiektu *Customer*, przypisywany jest model powiązany z tabelą bazy danych o nazwie *customers*. W linii 5 do modelu *Customer* przypisywana jest nowa wartość pola *item*, oraz wywołanie asynchronicznej funkcji *save()*, wykonującej odpowiedni zapis już w bazie danych.

#### 4.4. Aplikacja trenera (Marcin Olszewski)

<https://github.com/RunandPL/Frontend>

Zgodnie z założeniami, dokonanymi podczas analizy wymagań, aplikacja dla trenera będzie modulem webowym. Jej interfejs będzie dostępny z poziomu przeglądarki Internetowej. Dzięki zapewnieniu pełnego dostosowania do rozdzielczości przeglądarki(ang. *responsive*), zachowana zostanie wygoda korzystania z aplikacji również na urządzeniach mobilnych. Frameworkiem, którego użyjemy do implementacji aplikacji będzie AngularJS, otwarta biblioteka języka JavaScript, która jest wspierana i firmowana przez Google. Dzięki takiej decyzji, w naturalny sposób będziemy pracować w oparciu o wzorzec architektoniczny MVC(ang. Model-View-Controller). Szeroko stosowany od lat 70. minionego stulecia, jednak w programowaniu sieciowym został wprowadzony stosunkowo niedawno. Podstawą MVC jest zastosowanie wyraźnej separacji między logiką aplikacji(kontrolerem), zarządzaniem danymi(model) oraz sposobem prezentacji danych(widok).

**AngularJS** (<https://angularjs.org/>)

Możliwości w zakresie tworzenia aplikacji sieciowych są bardzo duże, a świadczy o tym chociażby szeroki przekrój technologii takich jak: PHP, Rails, Java EE, .NET czy Scala. Niestety często



możemy się przekonać, że równie wysoki jest stopień skomplikowania, który jest związany z procesem wytwarzania takich aplikacji. Technologia AngularJS powstała w celu ułatwienia programistom tworzenia aplikacji AJAX(ang. *Asynchronous JavaScript and XML*). Skuteczność AngularJS poparta jest doświadczeniem zespołu programistów Google, którzy zdobywali je pracując przy takich projektach jak Gmail, Mapy oraz Kalendarz. Dzięki uproszczeniu wykonywania niektórych czynności, jak np. wysyłanie żądań http, możemy zwrócić uwagę na decyzje projektowe, ułatwiające testowanie, dalszą rozbudowę aplikacji i jej konserwację. Dzisiaj, projekt AngularJS, rozwijany jest przez społeczność *open source* z całego świata.

W naszym projekcie zdecydowaliśmy się odejść od łączenia kodu HTML z danymi po stronie serwera i przekazywania wygenerowanej strony przeglądarce Internetowej. Zamiast tego, korzystamy z szablonów stron, które łączone są z danymi dzięki bibliotece AngularJS, a następnie przekazywane do przeglądarki. W ten sposób, rola serwera ogranicza się do przechowywania i

udostępniania zasobów statycznych szablonom oraz przekazywania danych niezbędnych do wypełnienia wspomnianych szablonów. Jest to podejście, które może się kojarzyć z aplikacjami, które według podejścia AJAX, posiadają jedną stronę, która wypełniana jest dynamicznie danymi.

Zgodnie z założeniem implementacji architektury MVC, w aplikacjach AngularJS widokiem jest DOM(ang. *Document Object Model*) – obiektowy model dokumentu, kontrolerami są klasy JavaScript, a dane modelu przechowywane są we właściwościach obiektu. Dzięki tak dużej separacji warstw, możliwe staje się dokładne pokrycie aplikacji testami, co przy innym podejściu, dokładając do tego złą strukturę kodu, było wręcz niemożliwe.

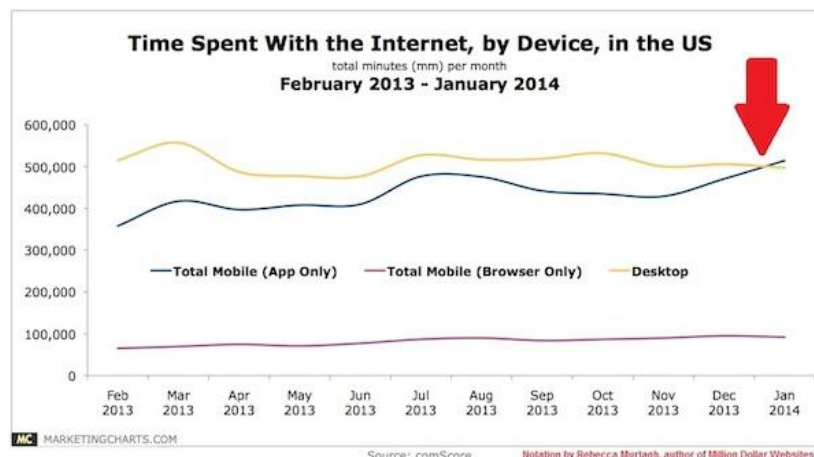
Dużo mówi się o rozszerzaniu możliwości istniejących już rozwiązań. Angular rozszerza możliwości HTML poprzez wprowadzenie dwukierunkowego wiązania danych(ang. *Two Way Data-Binding*). Jak to działa? Stworzone w HTMLu szablony łączone są zgodnie z danymi zawartymi w zakresie(ang. *scope*) zdefiniowanym przez model. Serwis \$scope w Angular identyfikuje zmiany w modelu, po czym modyfikuje HTML w widoku poprzez kontroler. Analogicznie, wszelkie zmiany w widoku są widoczne w modelu. Pozwala to ominąć potrzebę manipulowania na drzewie DOMu i znacznie przyspiesza tworzenie aplikacji Internetowych. Biblioteka dostarcza ponadto wiele przydatnych rozwiązań, które dotychczas musieliśmy implementować sami. Są to m.in. mechanizmy routingu, wykorzystanie części stron(ang. *partial*), czy wstrzykiwanie zależności i wykorzystanie dyrektyw. Równie proste staje się wysyłanie żądań http. Poniżej został przedstawiony przykładowy kod najprostszej aplikacji MVC, opartej o bibliotekę AngularJS.

```
1.      <script type="text/javascript">
2.          var myApp = angular.module('myApp', []);
3.          myApp.ExampleController = function ($scope) {
4.              $scope.name = 'RunAnd';
5.          };
6.      </script>
7.
8.      <div ng-app="myApp" ng-controller="ExampleController">
9.          <h1>Aplikacja: {{ name }}!</h1>
10.     </div>
```

**Bootstrap** (<http://getbootstrap.com/>)

Projektując, a następnie implementując aplikacje dla użytkowników z graficznym interfejsem użytkownika(ang. *Graphical User Interface, GUI*), musimy zadbać nie tylko o to, aby spełniały swoje podstawowe funkcje, lecz dobrze zastanowić nad tym kto i na jakich urządzeniach będzie korzystał z naszych produktów.

Według firmy *Incisive Media* (dostawcy informacji biznesowych), która porównała łączny czas dostępu do Internetu na urządzeniach mobilnych oraz na komputerach osobistych od lutego 2013 roku do stycznia 2014 roku, nastąpił już moment, kiedy użytkownicy Internetu korzystają z niego więcej za pośrednictwem urządzeń mobilnych. Wyraźny jest też wzrost czasu korzystania z Internetu za pośrednictwem przeglądarek na urządzeniach mobilnych.



<http://cms.searchenginewatch.com/IMG/303/293303/time-spent-on-Internet-by-device-in-us.jpg?1404760136>

Zdecydowaliśmy się zatem na wykorzystanie w aplikacji trenerskiej, rozwijanego przez programistów Twittera, Framework CSS o nazwie Bootstrap. Zawiera on zestaw narzędzi, które ułatwiają tworzenie interfejsu graficznego aplikacji Internetowych. Bazuje m.in. na gotowych rozwiązaniach HTML oraz CSS. Dzięki wykorzystaniu bootstrapa i jego klas, otrzymamy interfejs dostosowany do urządzenia, nie zależnie czy jest to komputer PC, tablet, czy telefon komórkowy. Wyświetlane moduły dostosują swoją wielkość i wygląd do rozdzielczości urządzenia. Strony i aplikacje posiadające takie właściwości możemy określić mianem responsywnych(ang. *responsive*).



Bootstrap może być stosowany m.in. do stylizacji formularzy, przycisków, menu i wielu innych wyświetlanych. Framework wykorzystuje także język JavaScript.

Aby zintegrować aplikację z Bootstrapem wystarczy pobrać ze strony projektu skompilowany zbiór arkuszy CSS, bibliotekę JavaScript oraz czcionki, a następnie załączyć pobrane zasoby na danej stronie html.

```

1. <!-- Bootstrap core CSS -->
2. <link href="bootstrap/dist/css/bootstrap.css" rel="stylesheet">
3. <!-- Bootstrap theme -->
4. <link href="bootstrap/dist/css/bootstrap-theme.css" rel="stylesheet">

```

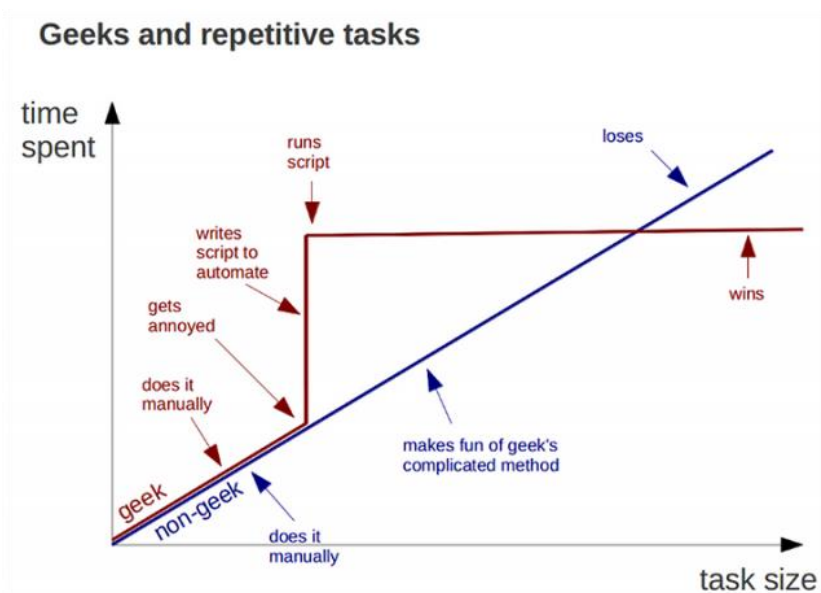
Wspierane przeglądarki:

	Chrome	Firefox	Internet Explorer	Opera	Safari
<b>Android</b>	Supported	Supported	N/A	Not Supported	N/A
<b>iOS</b>	Supported	N/A*		Not Supported	Supported
<b>Mac OS X</b>	Supported	Supported		Supported	Supported
<b>Windows</b>	Supported	Supported	Supported	Supported	Not Supported

N/A – brak odpowiedzi

**Bower** (<http://bower.io/>)

Wytwarzanie oprogramowania składa się w dużej mierze z czynności powtarzalnych, takich jak minimalizacja skryptów JS lub kompilacja stylów SASS do CSS. Są to często wykonywane czynności, które zabierają programiście wraz z upływem czasu i rozrastaniem się projektu, coraz więcej czasu. Zależności te przedstawia poniższy wykres:



Rys.3. <http://webmastah.pl/wp-content/uploads/2013/12/lazy-graph-640x457.png>

W naszym przypadku, najwięcej czasu zajmowałoby wyszukiwanie oraz instalacja bibliotek JS, a w trakcie utrzymania projektu, ich aktualizacja. Warto jednak wykorzystać dostępny manager pakietów Bower, którego zadaniem jest zarządzanie wszystkimi bibliotekami, które będziemy wykorzystywali w przeglądarce. Jest to dojrzały projekt, a świadczy o tym liczba ponad 6000 różnych bibliotek, które posiada. Jego użycie jest bardzo proste i ogranicza się do znajomości 7 podstawowych poleceń.

```
7. npm install -g bower
8. bower init
9. bower search [<name>]
10. bower info [<name>]
11. bower install [<name>]
12. bower update [<name>]
13. bower uninstall [<name>]
```

W pierwszej linii listingu znajduje się polecenie instalacji bowera, następnie jego inicjalizacja. Dostępność interesującej nas biblioteki sprawdzamy poleceniem z linii nr 3, gdzie jako [<name>] należy wprowadzić nazwę biblioteki, np. jquery.

Polecenie z linii nr 5 pozwoli nam zainstalować bibliotekę, a aktualizację wykonamy wydając polecenie z linii 6.

Domyślną lokalizacją dla bibliotek Bowera jest folder /bower\_components, jednak można to zmienić w pliku konfiguracyjnym .bowerrc. Kolejnym ważnym plikiem jest bower.json, który przechowuje informację o potrzebnych bibliotekach oraz ich wersji. Plik bower.json można wygenerować poleceniem z linii nr 2 lub można stworzyć go ręcznie.

```
1. {
2.   "name": "runand-frontend",
3.   "description": "A RunAnd AngularJS Project",
4.   "version": "0.2.0",
5.   "homepage": "https://github.com/RunandPL",
6.   "license": "MIT",
7.   "private": true,
8.   "dependencies": {
9.     "angular": "1.2.x",
10.    "angular-route": "1.2.x",
11.    "angular-loader": "1.2.x",
12.    "angular-mocks": "~1.2.x",
13.    "html5-boilerplate": "~4.3.0",
14.    "bootstrap": ">= 3.0.0",
15.    "angular-google-maps": "~2.0.6"
16.  }
17. }
```

## Angular Google Maps (Sebastian Miałkowski)

<http://angular-ui.github.io/angular-google-maps/>

W naszej pracy inżynierskiej, zarówno aplikacji mobilnej oraz webowej, korzystamy z dostępu do map. Zdecydowaliśmy się na usługi oferowane przez firmę Google, ze względu na jej renomę oraz jakość świadczonych usług. W dalszej części rozdziału opiszę sposób rejestracji, uwierzytelniania oraz komunikacji z serwisami Google. Są one dostępne dla każdego programisty, który zechce z nich skorzystać. Jedyne co musi zrobić to utworzyć konto Google a następnie zarejestrować tworzoną przez siebie aplikację. Kiedy to zrobimy dostaniemy dostęp do wersji bezpłatnych serwisów, w tej wersji większość z nich daje nam możliwość wykonania określonej, dziennej liczby zapytań. Na czas rozwoju oraz testów naszej aplikacji jest to wystarczająca opcja, gdyż liczba zapytań sięga zazwyczaj kilku tysięcy. Gdy jednak przestanie nam to wystarczać, pozostaje nam konieczność ponoszenia opłat dodatkowych opłat za dodatkowe zapytania.

Proces rejestracji jest niezwykle prosty i szybki, zajmuje dosłownie kilka minut. Jedyne co nam będzie potrzebne to nazwa pakietu, w którym znajduje się nasza aplikacja oraz wygenerowany dla niej (odcisk?)certyfikatu. Kiedy wszystko będzie gotowe należy wejść na stronę Google Developer Console. Z jej poziomu mamy możliwość zarządzania zarejestrowanymi aplikacjami, włączonymi dla naszych aplikacji serwisami, przeglądania płatności oraz statystyki. Rejestrację aplikacji



rozpoczynamy od rozwinięcia zakładki „APIs & auth” a następnie kliknięcia pozycji „Cridentials”. W nowym oknie wybieramy opcję „Create new Key”. Dalej kierujemy się instrukcjami podanymi na ekranie, ich wynikiem będzie rejestracja naszej aplikacji. Następnie należy stworzyć identyfikator protokołu OAuth 2.0. Google używa go do uwierzytelniania oraz autoryzacji usług i aplikacji. Proces tworzenia identyfikatora jest równie prosty jak rejestracji aplikacji.

Google udostępnia do naszego użytku, odpłatnego lub darmowego, ponad 70 usług. Domyślnie wszystkie są dla nas wyłączone przez co nasz aplikacja nie będzie mogła z nich skorzystać. W zakładce „APIs” znajdziemy listę wszystkich interfejsów, wybrane z nich mamy możliwość aktywować. Na liście prócz nazw interfejsów znajduje się też limit zapytań jaki możemy wykorzystać, a po aktywacji interfejsu mamy możliwość śledzenia ich wykorzystania.

## 5. IMPLEMENTACJA PROJEKTU

### Implementacja Google Maps API

#### 5.1. *Aplikacji mobilnej*(Sebastian Miałkowski)

Zgodnie z założeniami projektu, aplikacja mobilna została zaimplementowana na telefony z systemem Android. Zdecydowaliśmy się na to z dwóch powodów. Pierwszym była chęć lepszego poznania systemu oraz kontynuowanie nauki pisania aplikacji dla niego przeznaczonych, którą rozpoczęliśmy na przedmiocie Programowanie aplikacji mobilnych. Kolejnym, chęć dotarcia z naszą aplikacją do jak największej liczby odbiorców. Biorąc pod uwagę te dwa założenia, wybór systemu Android był wręcz oczywisty, gdyż jak wynika z tabeli [NR TABELI] jest to obecnie najpopularniejszy system na urządzenia mobilne, a jego udział w tym rynku stale rośnie.

Okres	Android	IOS	Windows Phone	BlackBerry OS	Inne
Połowa roku 2014	84.4%	11.7%	2.9%	0.5%	0.6%
Połowa roku 2013	81.2%	12.8%	3.6%	1.7%	0.6%
Połowa roku 2012	74.9%	14.4%	2.0%	4.1%	4.5%

Tabela [NR] Udział systemów mobilnych w rynku

(źródło: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>)

## 6.



Dodatkowo zdecydowaliśmy że najniższą wersją systemu wspieraną przez naszą aplikację będzie Android 4.1.2(Jelly Bean). Na podstawie tabeli [NR TABELI] widać że nasza aplikacja będzie wspierać 82.6% urządzeń z systemem firmy Google.

4.1.x	Jelly Bean	16	21.3%
4.2.x	Jelly Bean	17	20.4%
4.3	Jelly Bean	18	7.0%
4.4	KitKat	19	33.9%

Tabela [NR] Udział rynkowy poszczególnych wersji systemu Android

(źródło: [developer.android.com/about/dashboards/index.html](http://developer.android.com/about/dashboards/index.html))

Podczas implementacji aplikacji korzystaliśmy z dwóch frameworków: Butterknife oraz Picasso. Niezbędna również była możliwość wyświetlania map, by to uzyskać zintegrowaliśmy aplikację z serwisem Google Maps.

**Picasso**(<http://square.github.io/picasso>)

W czasach gdy serwisy społecznościowe cieszą się dużą popularnością, ludzie robią oraz wysyłają na nie coraz więcej zdjęć. Skutkuje to również tym że duża liczba aplikacji musi mieć zaimplementowaną możliwość ich obsługi. Poza tak oczywistymi przykładami jak *Facebook*, czy *Instagram* robienie oraz podgląd zdjęć udostępniają aplikacje treningowe. W takich właśnie aplikacjach zastosowanie znajduje *framework Picasso*. Zdjęcia czy obrazki często mają rozmiary po kilka megabajtów, ich wczytanie więc może powodować blokowanie interfejsu użytkownika. By temu zapobiec takie funkcje można zaimplementować na osobnych wątkach, jednak są to często operacje kłopotliwe oraz występujące w wielu miejscach. By ułatwić sobie pracę można skorzystać z Picasso, który wszystkie operacje wykona za nas a dodatkowo zajmie się obsługą błędów. Framework pozwala wyświetlenie obrazków pobranych z Internetu, wczytanych z dysku oraz pamięci. Prócz samego odczytu plików graficznych została nam udostępniona również możliwość ich prostej edycji np. zmiana rozmiaru, by lepiej mogły się dopasować wyglądu naszej aplikacji. Na wypadek gdyby dostępne możliwości edycji były dla nas niewystarczające, możemy zaimplementować własne i przekazać do wykorzystania dla Picasso. W naszej aplikacji używamy go do pobrania obrazka trasy, który jest generowany przez Google Static Map. Poniżej zamieszczam przykładowy fragment kodu powodujący pobranie obrazka z danego adresu Internetowego, a następnie załadowanie go do widok *imageView*.

```
1 Picasso.with(context).load("http://i.imgur.com/Dvpvklm").into(imageView);
```

**Uwierzytelnianie Google+ (Mateusz Pakulski)**

Przed rozpoczęciem integracji funkcji Google+ w aplikacjach konieczne jest utworzenie projektu w Google Developer Console i zainicjowanie GoogleApiClient w tworzonej aplikacji. Potrzebne biblioteki Google Play Services można pobrać z SDK Managera. Urządzenie do tworzenia i testowania ze względu na usługi Google Play Services może zostać zainstalowane tylko na emulatorach z Android Virtual Device bazujących na wersji 4.2.2 lub nowszej. Dokładniejszy opis konfiguracji Google Developer Console znajduje się w rozdziale [sebastiana](#) [tu](#) opis numer czy cokolwiek rozdziału [sebastiana](#)

Logowanie użytkowników w systemie android składa się z następujących etapów:

1. Przy uruchomieniu aplikacji łączy się ona z serwerem Google i sprawdza czy użytkownik jest zalogowany.
2. Jeżeli tak wywołuję metodę *onConnected()*.
3. Jeżeli nie wywołuję metodę *connectionFailed()* które zwraca obiekt *connectionResult* i ustawia flagi.
4. Gdy użytkownik kliknie zaloguj wywoływana jest metoda *ResolveError()* rozwiązująca błędy z obiektu *connectionResult* . Wywoływany jest ekran uprawnień (ang. consent screen) i ekran wyboru konta (ang. account chooser).
5. Gdy użytkownik zaakceptuje warunki zostaje zalogowany po czym wywoływana jest metoda *reconnect()* i powrót do punktu pierwszego.

Rozpoczęcie wykonywania metody *onConnected()* jest równoznaczne z uzyskaniem z serwera Google dostępu do danych użytkownika, które można potem w prosty sposób wyłuskać z obiektu *mGoogleApiClient* za pomocą klas i metod zawartych w pakiecie *com.google.android.gms.plus* . Do prawidłowego działania usług Google Play niezbędna jest deklaracja w aktywności wykorzystującej dane usługi dodatkowych zmiennych i obiektów służących jako globalne flagi lub komunikatory dla metod wykorzystywanych przez te usługi. Podczas implementacji autoryzacji i logowania niezbędne było wykorzystanie obiektów *mGoogleApiClient* typu *GoogleApiClient* i *mConnectionResult* typu *ConnectionResult* oraz dwóch flag logicznych *mIntentInProgress* oraz *mSignInClicked*.

#### 6.1. **Implementacja serwera** (Paweł Mazurek)

##### **Środowisko pracy**(<https://netbeans.org/>)

Serwer został implementowany w Netbeans IDE, zintegrowanym środowisku programistycznym napisanym w języku Java. Nie wspiera on bezpośrednio Node.js, ale posiada wygodny edytor Javascript, kolorujący składnie. Niestety nie posiada on debugera

Node.js, co wymusiło na mnie potrzebę używania logów konsolowych, w celach naprawiania błędów.

### Struktura projektu

/	-> główny folder aplikacji
node_modules/	-> moduły zarządzane przez manager pakietów NPM
src/	-> folder zawierający kody źródłowe
config/	-> folder zawierający plik konfiguracyjny
Configuration.js	-> moduł odpowiedzialny za konfigurację
public/	-> folder zawierający pliki udostępniane publicznie (zdjęcia)
Image.js	-> moduł odpowiedzialny za zapisywanie i udostępnianie zdjęć
LiveWorkout.js	-> moduł odpowiedzialny za zarządzanie treningami na żywo
Route.js	-> moduł odpowiedzialny za zarządzanie trasami
TrainerRunnerRequest.js	-> moduł zarządzający systemem wysyłania, odbierania, akceptacji i odrzucania zapytań
User.js	-> moduł zarządzający kontami użytkowników
Workout.js	-> moduł zarządzający treningami
WeatherService.js	-> moduł zarządzający danymi pogodowymi
Server.js	-> punkt startowy aplikacji, w którym zdefiniowane są wszystkie możliwe zapytania
package.json	-> plik konfiguracyjny managera pakietów NPM
weather.js	-> plik zawierający informacje pogodowe przechowywane na serwerze (serwer cyklicznie go odświeża)

### Uwierzytelnianie z tokenem

W rozdziale 4.3 opisałem na czym polega REST. Jednym z najważniejszych aspektów tego wzorca jest bezstanowość komunikacji klienta z serwerem. Aby to osiągnąć zdecydowałem się implementację uwierzytelniania przy użyciu tokenów.

Osiągnąłem to z pomocą JWT, tj. JSON Web Token. Jest to kompaktowy i bezpieczny sposób przesyłania informacji uwierzytelniających między serwerem, a klientem. Przepływ aktywności komunikacji przy jego użyciu wygląda następująco:

- Użytkownik wysyła serwerowi swoje dane, tj. nazwę użytkownika i hasło,
- Serwer sprawdza czy użytkownik istnieje,
- Jeżeli tak, to serwer szyfruje wszelkie potrzebne informacje o użytkowniku za pomocą swojego własnego klucza prywatnego (dane profilowe użytkownika) oraz czas wygasania tokena w ciąg znaków:
  - Do szyfrowania korzysta z wybranego algorytmu szyfrującego, domyślnie jest to HS256
  - Pierwsza część tokenu to informacja o wykorzystanym algorytmie szyfrującym, oraz typie tokenu:

```
{  
  "alg": "HS256",  
  "typ": "JWT"
```

- ```

    }
  
```
- Druga część tokenu to informacje profilowe i czas wygasania, np:
 

```

{
  "name": "John Doe",
  "admin": true,
  "exp": 124523
}
      
```
  - Trzecia część, to tzw. Payload, funkcja skrótu, która służy do sprawdzenia integralności tokenu.
  - Po zaszyfrowaniu token może wyglądać następująco:
 

```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
.eyJzdWiiOiJlYmZQ1Njc4OTAsIm5hbWUiOiJKb2hulERvZSIsImFkbWluljp0cnVlfQ
.eoaDVGTCIRdfxUZXiPs3f8FmJDkDE_VCQFXqKxpLsts
      
```

Poszczególne części tokenu są rozdzielone kropkami. Gdy serwer go wygeneruje, to wysyła go użytkownikowi.
  - Użytkownik jeżeli chce otrzymać dostęp do zasobu chronionego, musi wysłać ten token razem z zapytaniem, jako nagłówek, o kluczu Authorization, i wartości Bearer <<token>>.
  - Serwer otrzymując token, deszyfruje go, używając własnego klucza prywatnego. Następnie może odczytać z niego informacje o użytkowniku, np. jakie ma uprawnienia, jak się nazywa itp.

Dzięki takiej implementacji, uwierzytelnianie nie wymaga przechowywania po stronie serwera żadnych informacji o zalogowanych użytkownikach, komunikacja (zgodnie z wzorcem REST) jest kompletnie bezstanowa. Klasyczne podejścia do uwierzytelniania, wymagają użycia mechanizmów sesji, czyli przechowywaniu w pamięci serwera informacji o aktualnie zalogowanych użytkownikach.

Z uwierzytelniania z tokenem korzystają wszyscy klienci, czyli aplikacja Internetowa, i aplikacja mobilna.

## Serwis pogodowy

Serwer udostępnia dane pogodowe dla terenu Polski. Aby to zrealizować, pobiera je z zewnętrznego serwisu o nazwie Forecast.io. Udostępnia on wykonanie darmowego tysiąca zapytań o pogodę, wystarczy posiadać token, który otrzymujemy po rejestracji. Wysyłając dostarczony token, oraz współrzędne geograficzne badanego miejsca, otrzymamy w odpowiedzi plik JSON zawierający informacje pogodowe na najbliższe 24 godziny. Aby uniezależnić się od limitu 1000 zapytań dziennie, i ominąć ograniczenia darmowej wersji serwisu, opracowałem własny algorytm przechowujący pobrane dane pogodowe w pamięci, i udostępniające je na zewnątrz.

Podzieliłem obszar geograficzny Polski na siatkę, widoczną na rysunku nr [NR RYSUNKU].



*Rys.[NR] Polska podzielona siatką*

Ilość krutek w poziomie i w pionie jest w pełni konfigurowalna. Po wykonaniu takiej siatki, serwer, co określony interwał czasowy wysyła do Forecast.io zapytania o pogodę, jako współrzędne podając środek każdej z krutek. Otrzymane dane zapisuje najpierw do pamięci podręcznej, następnie do pliku. Gdy serwer zostaje zrestartowany, sprawdza najpierw datę modyfikacji pliku pogodowego, jeżeli dane są aktualne, to odczytuje je z pliku do pamięci podręcznej. Jeżeli serwer był wyłączony przez długi czas, i dane pogodowe się dezaktualizowały, to generuje nowy plik, zastępując poprzedni, pobierając aktualne dane pogodowe.

Jeżeli użytkownik wykonuje zapytanie o dane pogodowe, spoza wspieranego aktualnie obszaru Polski, serwer zwraca mu komunikat, o braku wsparcia informacji pogodowej dla tych współrzędnych.

### **Treningi na żywo**

Treningi prowadzone przez zawodników, nie są przechowywane w bazie danych, tylko w pamięci podręcznej. Zdecydowałem się na taki krok, z powodów optymalizacyjnych. Podczas testów, średni czas odpowiedzi serwera, przy odczycie z bazy danych, oscylował w granicach 400ms, podczas gdy serwer nie korzystał z bazy danych, a informacje trzymał w pamięci, to czas odpowiedzi to już okolice 50ms. Ma to spore znaczenie w przypadku treningów na żywo. Ponieważ po rozpoczęciu treningu, dane aktualizowane są na bieżąco, co krótki okres czasu do serwera wysyłane jest zapytanie aktualizujące trening. Dlatego w takim przypadku nie można sobie pozwolić na długie czasy odpowiedzi serwera.

### **Zapisywanie i udostępnianie zdjęć tras**

Biegacze mają możliwość zapisania zdjęcia trasy na serwerze, łącznie z ich lokalizacją. Zdjęcia po stronie serwera odbierane są w kodowaniu base64. Zapisuje on je w systemie plików,

nadając im unikalną nazwę, będącą połączeniem aktualnej daty, fragmentu obrazka, oraz losowej liczby. Ścieżka do obrazka, oraz współrzędne zapisywane są w bazie danych.

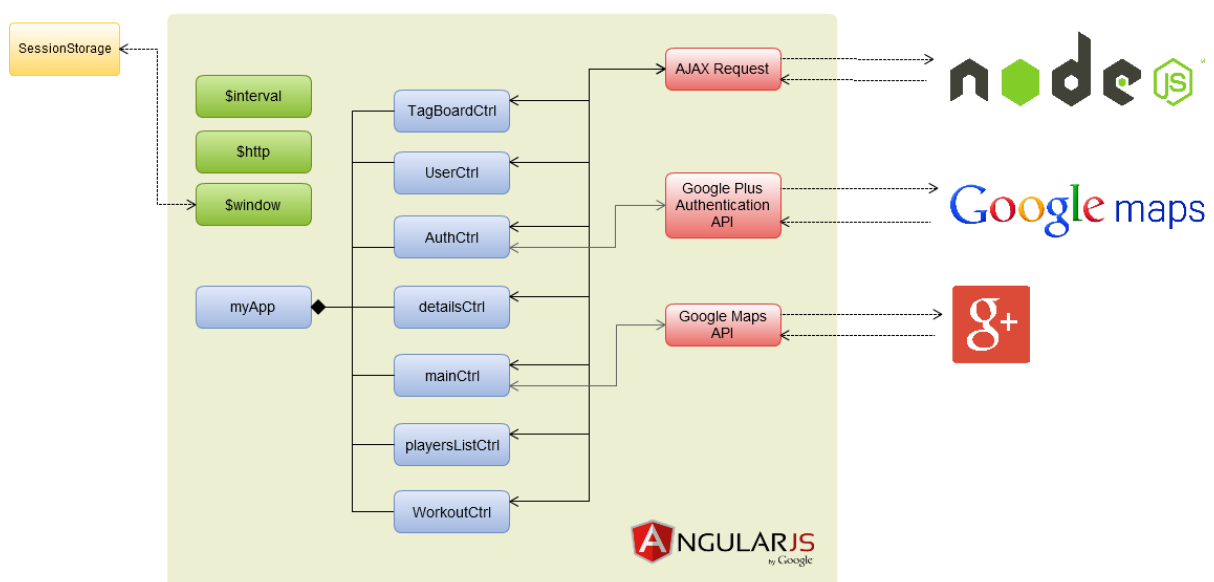
Zapis obrazków zrealizowany jest w taki sposób, aby uniknąć bezpośredniego przechowywania całych obrazków w bazie danych, co jest mało wydajne. Zdjęcia w bazie danych powiązane są bezpośrednio z trasami, dlatego w odpowiedzi na zapytanie pobrania tras, serwer zwraca listę adresów do zdjęć zrobionych na danej trasie.

### Wysyłanie, akceptacja i odrzucanie zapytań o dodanie trenera

Projekt opiera się na współdziałaniu trenera i zawodnika. Dlatego bardzo istotną rzeczą było zaimplementowanie sposobu dopisywania trenera do zawodnika. Rozwiązaliśmy to w następujący sposób. Trener w aplikacji Internetowej, ma możliwość wysłania zapytania o dodanie zawodnika, podając nazwę użytkownika docelowego. Zawodnik, korzystając z aplikacji mobilnej może pobrać listę zapytań wysłanych do niego, łącznie z numerem danego zapytania. Gdy uzyska już listę zapytań, ma możliwość akceptacji, lub odrzucenia ich, przy wykorzystaniu odpowiedniego numeru zapytania.

Tego rodzaju dwustronna akceptacja jest najlepszą możliwą opcją wykonania tej funkcjonalności. W przypadku gdyby dodawanie trenera do zawodnika było operacją jednostronną, zawodnik nie miałby prawa głosu, co nie wydaje się dobrym rozwiązaniem.

## 6.2. Implementacja aplikacji WWW (Marcin Olszewski)



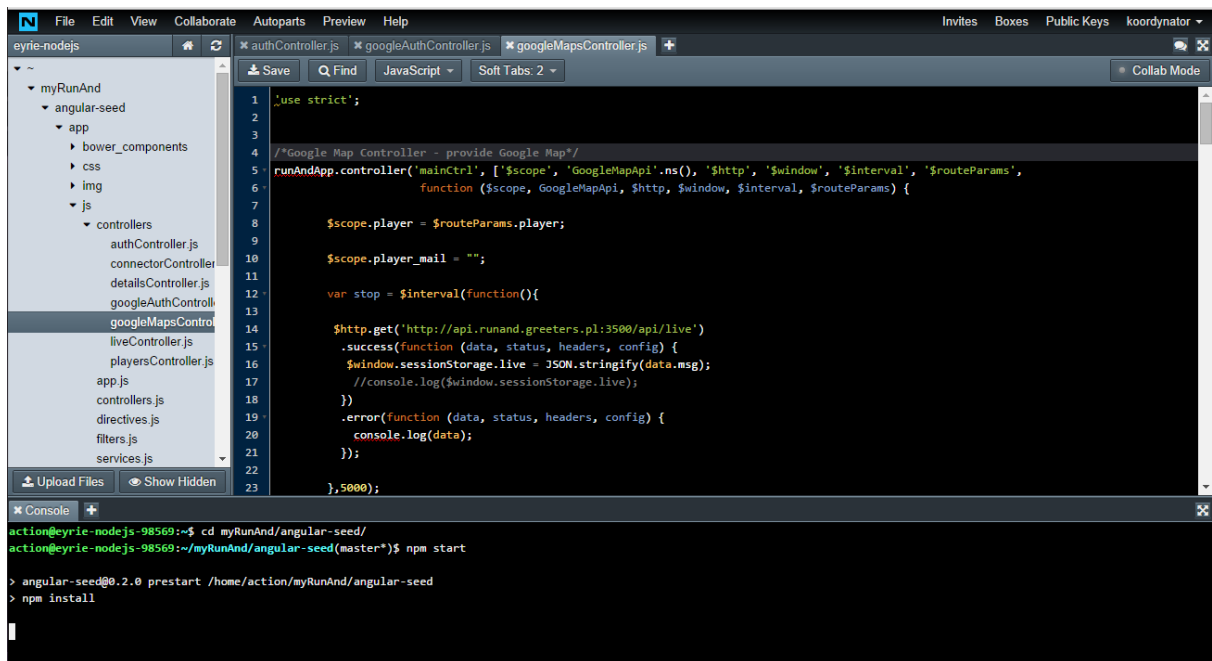
Powyższy diagram przedstawia strukturę modułów aplikacji trenerskiej. Aplikacja AngularJS (na diagramie *myApp*), zawiera kontrolery, które komunikują się asynchronicznie poprzez żądania AJAX z serwerem, udostępniającym usługi REST. AngularJS umożliwia wysyłanie żądań poprzez wykonywanie metod obiektu `$http` (metody `.post()` oraz `.get()`), które poprzez mechanizm Dependency Injections, wstrzykujemy do kontrolera. Wyróżnione zostały także obiekty `$interval` – odpowiadający za cykliczne wywoływanie funkcji oraz `$window` – dzięki niemu uzyskujemy dostęp do obiektu *window* języka JavaScript i możemy zapisywać i odczytywać dane z `SessionStorage`. Aplikacja składa się z 7 kontrolerów:

- TagBoardCtrl
- UserCtrl
- AuthCtrl
- detailsCtrl
- mainCtrl
- playersListCtrl
- WorkoutCtrl

W projekcie użyte zostały także niestandardowe moduły do obsługi Google Maps API oraz uwierzytelniania Google Plus.

### **NitrousIO i struktura projektu**

Jedną z głównych decyzji projektowych był wybór środowiska implementacji aplikacji trenera. AngularJS wydał nam się najlepszym rozwiązaniem, ponieważ dostarcza on najpotrzebniejszych mechanizmów, takich, jak: tworzenie szablonów widoków, routing URL, wiązanie danych do kontrolek widoku oraz składowanie danych, czyli wsparcie dla technologii AJAX, która była niezbędna do realizacji komunikacji z serwerem poprzez usługę REST. Aby umożliwić hosting aplikacji trenerskiej, wykorzystaliśmy środowisko *Nitrous* udostępniające maszynę wirtualną w chmurze, która wspiera takie platformy, jak Ruby, Python oraz Node.js. Tworząc darmowe konto w serwisie Nitrous, otrzymaliśmy dostęp do maszyny wirtualnej posiadającej 300 MB pamięci RAM oraz 1000 MB pamięci dyskowej. Dodatkowo, zyskaliśmy dostęp do narzędzi programistycznych – IDE z poziomu przeglądarki Internetowej, dzięki czemu, w tym samym czasie kilku użytkowników może edytować ten sam kod. IDE posiada także konsolę, która przez SSH łączy się z serwerem, oraz chat do komunikacji programistów.



Kolejnym krokiem, było utworzenie struktury projektu. Jako, że aplikacja oparta o Framework AngularJS umożliwia korzystanie z kontrolerów, szablonów widoku, dyrektyw, serwisów oraz zawiera zasoby statyczne (np. pliki json, arkusze stylów, pliki graficzne), należało utworzyć strukturę projektu, która w intuicyjny sposób zorganizuje projekt.

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| app/                | -> główny folder aplikacji                              |
| bower_components/   | -> dodatkowe biblioteki menedżera pakietów bower        |
| css/                | -> arkusze stylów css                                   |
| img/                | -> pliki graficzne                                      |
| js/                 |                                                         |
| controllers/        | -> kontrolery komunikujące się z serwerem               |
| controllers.js      | -> zbiór kontrolerów związanych z wyświetlaniem layoutu |
| app.js              | -> punkt wejścia do aplikacji AngularJS                 |
| directives.js       | -> dyrektywy dostarczające zawartość szablonów          |
| services.js         | -> usługi, np. interceptory                             |
| navbar_provider/    | -> pliki konfiguracyjne json, dostępne opcje w menu     |
| partials/           | -> pliki widoków aplikacji                              |
| tpl/                | -> pliki szablonów dla widoków                          |
| google_map.html     | -> główny widok mapy                                    |
| tag_board.html      | -> widok listy opublikowanych treningów                 |
| workouts.html       | -> widok listy aktualnie trenujących zawodników         |
| routes/             |                                                         |
| dummy_route.json    | -> zastępcza trasa testowa                              |
| slider/             |                                                         |
| slides.json         | -> plik konfiguracyjny slidera                          |
| workouts_provider/  |                                                         |
| dummy_workouts.json | -> dane testowe, lista opublikowanych tras              |
| index.html          | -> strona główna aplikacji                              |

## Widok - szablony i dyrektywy



Koncepcję wzorca MVC we frameworku AngularJS realizuje się poprzez składanie części widoku w jeden, który widzi użytkownik. W aplikacji trenerskiej zaimplementowane są tylko trzy główne partiale – widok mapy, listy opublikowanych tras oraz treningów. Żeby uprościć i uporządkować kod źródłowy partiali, można zastosować szablony. Tą funkcję Framework uzyskujemy poprzez mechanizm dyrektyw, który jest przez jego twórców nazwany poszerzeniem funkcji HTML.

Dyrektywy w kodzie HTML, to miejsca, w które Angular w fazie kompilacji, podczas skanowania drzewa DOM, podpiną nowe funkcje, które mają zastosowanie dla danego węzła oraz jego podwęzłów(tzw. dzieci). Można dzięki temu uzyskać komponenty, które pozwalają manipulować drzewem DOM oraz dołączać mu nowe funkcje. Przykładami predefiniowanych dyrektyw Framework są ng-app, ng-controller lub ng-repeat, która pozwala budować dynamiczną listę w oparciu o kolekcję obiektów.

Dyrektywy można tworzyć na kilka różnych sposobów, posiadają także szereg parametrów konfiguracyjnych. W poniższym przykładzie zaprezentowany jest przykład prostej dyrektywy, dołączającej plik szablonu do drzewa DOM.

```
myDirectives.directive('carousellDirective', function () {
    return{
        templateUrl: 'partials/tpl/carousell.tpl.html'
    }
});

<!-- użycie dyrektywy -->
<div class="wrap" navbar-directive></div>
```

Zastosowany w powyższym listingu parametr templateUrl pozwala na określenie ścieżki do szablonu – osobnego pliku HTML. Szablon taki zostanie załadowany asynchronicznie oraz będzie zapisany w pamięci podręcznej(ang. *cache*).

## Routing URL

W odniesieniu do aplikacji Internetowych, trasowanie(ang. *routing*) to mechanizm dynamicznego określania ścieżek URL(ścieżek dostępu, które są zawarte w URL żądania HTTP) odwzorowywanych dla różnych obszarów aplikacji. Routing w aplikacjach Internetowych pozwala utrzymać porządek po stronie programisty, ale z drugiej strony powoduje, że nasza aplikacja jest bardziej czytelna po stronie użytkownika. Pozbywamy się nieczytelnych i często długich ścieżek, jak np. `index.php?article_id=12&category=2` na rzecz czytelniejszego adresu, np. `/articles/run-and`. Dostarczenie mechanizmów routingu jest jedną z zalet korzystania z frameworków. Oprócz Symfony, Zend Framework, czy Backbone, również AngularJS je udostępnia. Należy jednak dołączyć do pliku `index.html` bibliotekę *angular-route.js* i w prosty sposób skonfigurować dostawcę routingu(ang. *routing provider*). Poniżej znajduje się pełna konfiguracja routingu dla frontendu aplikacji trenerskiej.

```

1.  /*deklaracja modułów wykorzystywanych w aplikacji AngularJS*/
2.  var myApp = angular.module('myApp', [
3.      'ngRoute', /*dołączenie modułu routingu do aplikacji myApp*/
4.      'myApp.filters',
5.      'myApp.services',
6.      'myApp.directives',
7.      'myApp.controllers',
8.      'google-maps'.ns(),
9.      'googleplus'
10.  ]);
11.
12.  /*konfiguracja route providera - dodawanie reguł routingu*/
13.  myApp.config(['$routeProvider', function ($routeProvider) {
14.      $routeProvider.when('/workouts_board', {
15.          templateUrl: 'partials/workouts_board.html',
16.          controller: 'WorkoutsBoardCtrl'
17.      });
18.      $routeProvider.when('/main_map', {
19.          templateUrl: 'partials/google_map.html',
20.          controller: 'WorkoutCtrl'
21.      });
22.      $routeProvider.otherwise({
23.          redirectTo: '/tag_board'
24.      });
25.  }]);

```

Z powyższego listingu pochodzącego z pliku app/js/app.js wynika, że korzystaliśmy z dwóch reguł obsługujących wyświetlanie tablicy z opublikowanymi treningami powiązanej z kontrolerem WorkoutsBoardCtrl oraz mapy, na której wykonywane są operacje właściwe dla trenera oraz śledzonego treningu (np. wyświetlanie trasy biegacza na żywo lub wysyłanie zlecenia treningu i listowanie zawodników trenowanych przez zalogowanego trenera), obsługiwanej przez WorkoutCtrl. Określiliśmy także routing domyślny, gdy adresu url nie będzie można dopasować do żadnej z reguł (wyświetli nam się tablica opublikowanych treningów) i szablony html, które zostaną wczytane w miejsce dyrektywy ng-view w pliku index.html.

```

1.  <div ng-view></div>

```

### Komunikacja z serwerem, interceptory, pobieranie danych

Warstwę danych w aplikacji trenerskiej możemy podzielić na zasoby statyczne oraz dynamiczne. Do danych statycznych zaliczają się pliki .json zawierające dane ładowane przez kontrolery, w przypadku, kiedy serwer nie odpowiada na żądania http lub zwraca nieprawidłowe dane lub dane komponentów takich jak menu oraz slider.

Są to między innymi pliki:

- dummy\_route.json – przykładowa trasa do zaznaczenia na mapie

- dummy\_workouts.json – przykładowe trasy opublikowane
- slides.json – plik konfiguracyjny rotatora bannerów
- open.json, trainer.json – pliki konfiguracyjne menu

Przykład obiektu pojedynczego slajdu dla rotatora bannerów znajduje się poniżej.

```
{
  "id":1,
  "class": "active",
  "title":"RunAnd",
  "content":"Pobierz aplikacje mobilną, dzięki której będziesz mógł
ułożyć swój własny plan treningowy. Podejmij wyzwanie już dziś!",
  "href":"#",
  "btttn":"Pobierz",
  "img":"/img/slider/3.png",
  "alt":"Obrazek1"
},
```

Zasoby statyczne znajdują się na serwerze, z którym komunikacja odbywa się poprzez usługi REST (ang. Representational State Transfer). Do danych, pobieranych z serwera należą:

- lista tras opublikowanych
- lista zawodników, przypisanych do trenera
- pobierana cyklicznie aktualna pozycja i szczegóły treningu
- lista aktualnie trenujących zawodników

W projekcie zrealizowaliśmy uwierzytelnianie z tokenem, który pobierany jest podczas logowania i pamiętany przez cały czas trwania sesji użytkownika – w obiekcie SessionStorage. Token ten musi zostać dołączony do każdego zapytania kierowanego do serwera. Aby zapobiegać powielaniu kodu, zaimplementowany został specjalny interceptor:

<pre>myServices.factory('authInterceptor', function (\$rootScope, \$q, \$window) {   return {     request: function (config) {       config.headers = config.headers    {};       if (\$window.sessionStorage.token) {         config.headers.Authorization = 'Bearer ' + \$window.sessionStorage.token;       }       return config;     },</pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<pre>         response: function (response) {             if (response.status === 401) {                 alert("401");             }             return response    \$q.when(response);         }     }; }); </pre>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Dodaje on specjalny nagłówek wymagany przy realizacji uwierzytelniania z tokenem – Authorization. Framework AngularJS udostępnia dedykowany obiekt w celu wysyłania żądań AJAX (POST, GET), do którego mamy dostęp w kontrolerze dzięki mechanizmowi Dependency Injections.

W powyższym listingu znajduje się implementacja kontrolera, do którego w parametrach wstrzyknięto obiekt \$http. W kontrolerze zaimplementowano pobieranie konfiguracji rotatora bannerów.

	<pre> runAndApp.controller('SliderCtrl', function (\$scope, \$http) {     \$http.get('slider/slides.json').success(function (data) {         \$scope.slides = data;     }); }); </pre>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7. REZULTATY PROJEKTU

## 8. PODSUMOWANIE

### 8.1. Wnioski

### 8.2. Perspektywy

### 8.3. Wdrożenie systemu na serwerze produkcyjnym (Paweł Mazurek)

**9. W TRAKCIE POWSTAWANIA PROJEKTU, Z INICJATYWY PROMOTORA, ZREZYGNOWALIŚMY Z WDROŻENIA SYSTEMU NA SERWERZE KATEDRALNYM. W RAMACH ZASTĘPSTWA, WDROŻYLIŚMY SERWER W TYM SAMYM ŚRODOWISKU, W KTÓRYM DZIAŁA PROJEKTOWY REDMINE. TO ZNACZY, NA MASZYNIE Z SYSTEMEM UBUNTU 14.04, NODE.JS W WERSJI 0.10.31 I SERWEREM POSTGRESQL. ABY RESTARTOWAĆ SERWER, W PRZYPADKU AKTUALIZACJI PLIKÓW, LUB MOŻLIWEJ AWARII, UŻYLIŚMY MODUŁU PM2 DO NODE.JS. W PRZYPADKU NIEOCZEKIWANEGO ZAKOŃCZENIA PRACY PROCESU, WSKRZESZA ON GO. JEŻELI PLIKI ZOSTANĄ**

**ZAKTUALIZOWANE, PROCES SERWERA JEST ODNAWIANY AUTOMATYCZNIE, BEZ OKRESU PRZEJŚCIOWEGO GDY SERWER JEST WYŁĄCZONY.**

API serwera jest dostępne pod adresem <http://api.runand.greeters.pl/> na porcie 3500.

## 10. WYKAZ LITERATURY

1. <https://angularjs.org/>
2. <http://bower.io/>
3. <http://angular-ui.github.io/angular-google-maps>
4. *AngularJS*, Brad Green, Shyam Seshadri, tłum. Robert Górczyński, wyd. Helion 2014
5. Winiarski R., Przewęda R., Wit B., Jegier A.: Sport dla wszystkich, TKKF Warszawa 1995
6. <http://www.redmine.org/>
7. Version Control with Git, 2nd Edition, Joe Loeliger, Matthew McCullough, tłum. Zdzisław Płoski, wyd. Helion 2012
8. Rebecca Murtagh, Mobile Now Exceeds PC: The Biggest Shift Since the Internet Began, <http://searchenginewatch.com/article/2353616/Mobile-Now-Exceeds-PC-The-Biggest-Shift-Since-the-Internet-Began>
9. Pro Android Apps Performance Optimization, Huerve Guihot, tłum. Tomasz Walczak, wyd. Helion 2013
10. The Java™ Native Interface Programmer's Guide and Specification, Sheng Liang, 1999 Sun Microsystems, Inc.
11. <http://nodejs.org/>
12. Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, [http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\\_dissertation.pdf](http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)
13. <http://blog.caustik.com/2012/08/19/node-js-w1m-concurrent-connections/>
14. <https://www.npmjs.org/>
15. <http://bookshelfjs.org/>
16. źródło: <https://developers.google.com/+>
17. <http://oauth.net/2/>

## **11. ZAŁĄCZNIKI**

### **11.1. Wykaz ilustracji**

### 11.2. Wykaz tabel

### **11.3. Wykaz listingów**

### **11.4. Podział prac**

#### **Marcin Olszewski**

Odpowiedzialny za konfigurację i administrację systemu zarządzania projektem Redmine, pełnił rolę koordynatora zespołu. Administrator repozytorium GitHub projektu. Przygotował projekt architektury, dokonał wyboru technologii oraz zaimplementował aplikację dla trenera.

Sebastian Miałkowski

Mateusz Pakulski

Paweł Mazurek

### **11.5. Instrukcja instalacji i konfiguracji systemu**

### **11.6. Instrukcja laboratoryjna** (Marcin Olszewski)