Rok akademicki 2011/2012

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

PRACA DYPLOMOWA INŻYNIERSKA

Kacper Mazur

# Użycie aplikacji mobilnej do monitorowania położenia dzieci

Opiekun pracy
mgr inż. Waldemar Grabski

Ocena: .....

.....

Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Specjalność: Inżynieria Systemów Informatycznych

Data urodzenia: 1989.09.26

Data rozpoczęcia studiów: 2008.10.01

Życiorys

Urodziłem się w Staszowie 26 września 1989 roku, a mieszkałem przez kolejne 19 lat w Sichowie Dużym, średniej wsi w województwie świętokrzyskim. Tam też uczęszczałem do szkoły podstawowej i gimnazjum, w którym to czasie zostałem laureatem olimpiady matematycznej. Ukończyłem Liceum Ogólnokształcące im. ks. kard. Stefana Wyszyńskiego w Staszowie.
W roku 2008 rozpocząłem studia na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na kierunku Informatyka. Od listopada 2011 podjąłem pracę jako młodszy programista .NET w firmie Business Management Software, która specjalizuje się w tworzeniu oprogramowania dla banków korporacyjnych.
Dziedzinami wiedzy które najbardziej mnie pasjonują na moim kierunku to aplikacje mobilne, użycie wzorców projektowych, nowości w dziedzinie oprogramowania (szczególnie platforma .NET), oraz stosowanie przejrzystego stylu implementacji, a także nowości w świecie sprzętu komputerowego i mobilnego.
Poza studiami i pracą lubię aktywnie spędzać czas (bieganie, pływanie, rolki, rower, a także sztuki walki), oraz obejrzeć dobry film lub pójść do teatru.

.....
Podpis studenta

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu ..... 20__ r

z wynikiem .....

Ogólny wynik studiów: .....

Dodatkowe wnioski i uwagi Komisji: .....

.....

.....

# STRESZCZENIE

Streszczenie pracy w języku polskim.

Głównym tematem pracy inżynierskiej jest przedstawienie sposobu tworzenia aplikacji mobilnej, na przykładzie urządzeń z systemem Windows Phone, oraz zaprezentowanie specyficznych problemów dla tego typu oprogramowania (jak zwrócenie uwagi na oszczędność baterii, użycie modułu GPS, czy przygotowanie interfejsu użytkownika na mniejszy ekran).
Etapy tworzenia oprogramowania są tutaj pokazane na przykładzie systemu do monitorowania położenia dzieci o nazwie Patronus. Proces przechodzi przez wszystkie etapy tworzenia aplikacji od porównania istniejących rozwiązań poprzez analizę dziedziny i projekt architektury systemu po implementację i testowanie.

Słowa kluczowe:

Windows Phone, monitorowanie położenia, urządzenia mobilne, .NET, Bing Maps

# Using the mobile application to supervise location of children

Summary in English.

The main topic of thesis is to show how to create a mobile application for devices with Windows Phone, and present specific problems for this type of software (such as saving the battery, using GPS, or preparing GUI for a smaller screen).
Stages of software development are shown here on an example of children position monitoring system called Patronus. The process goes through all stages of application development from a comparison of existing solutions, analysis and architecture design to implementation and testing.

Keywords:
Windows Phone, monitorowanie położenia, urządzenia mobilne, .NET, Bing Maps

# 1 Spis treści

1 Spis treści ..... 7
2 Wstęp ..... 11
3 Słownik pojęć ..... 13
4 Przegląd istniejących rozwiązań ..... 15
4.1 NaviTracker ..... 15
4.2 Navitracer ..... 15
4.3 Gdzie Jest Dziecko ..... 16
4.4 SpySat ..... 18
4.5 YourCargo ..... 19
4.6 Usługi oferowane przez operatorów sieci komórkowych ..... 20
4.7 Porównanie ..... 20
5 Analiza ..... 23
5.1 Analiza dziedziny ..... 23
5.2 Identyfikacja aktorów ..... 24
5.3 Wymagania funkcjonalne ..... 25
5.3.1 Konfigurowanie i zarządzanie systemem ..... 25
5.3.2 Konfiguracja urządzeń mobilnych ..... 26
5.3.3 Synchronizacja danych ..... 26
5.3.4 Monitorowanie położenia dziecka ..... 28
5.4 Wymagania niefunkcjonalne ..... 29
6 Architektura ..... 33
6.1 Część centralna – Patronus Serwer ..... 34
6.1.1 Baza danych ..... 34
6.1.2 Warstwa logiki ..... 38
6.1.2.1 Moduł synchronizacji danych ..... 38
6.1.2.2 Moduł usług sieciowych ..... 39
6.1.3 Warstwa prezentacji – Patronus Viewer ..... 46
6.1.3.1 Moduł operacji ..... 47
6.1.3.2 Algorytmy ..... 47
6.2 Aplikacja dziecka – Patronus Child ..... 51
6.2.1 Baza danych ..... 51
6.2.2 Warstwa logiki i operacji ..... 51
6.2.2.1 Usługi sieciowe ..... 52
6.2.2.2 Usługi działające w tle ..... 52

6.2.2.3 Moduł GPS ... 52
6.2.2.4 Moduł synchronizacji ... 52
6.2.2.5 Moduł sprawdzania pozycji ... 53
6.2.3 Warstwa prezentacji ... 54
6.3 Aplikacja rodzica – Patronus Parent ... 54
6.3.1 Baza danych ... 55
6.3.2 Warstwa logiki i operacji ... 55
6.3.2.1 Usługi w tle ... 55
6.3.2.2 Moduł synchronizacji ... 56
6.3.2.3 Moduł sprawdzania danych ... 56
6.3.3 Warstwa prezentacji ... 56
7 Implementacja ... 57
7.1 Środowisko pracy ... 57
7.1.1 Repozytorium kodu ... 57
7.1.2 Użyte technologie ... 58
7.1.3 Dodatki ... 59
7.1.4 Podział na projekty ... 59
7.2 Patronus Serwer ... 61
7.2.1 Model bazy danych ... 62
7.2.2 Warstwa logiki ... 62
7.2.2.1 Usługi sieciowe ... 63
7.2.3 Patronus Viewer ... 64
7.2.3.1 Kontroler ... 65
7.2.3.2 Mapy Bing ... 66
7.3 Aplikacje mobilne ... 67
7.3.1 Baza danych ... 68
7.3.2 Warstwa logiki ... 69
7.3.2.1 Usługi sieciowe ... 69
7.3.2.2 Usługi działające w tle ... 70
7.3.3 Patronus Parent ... 72
7.3.3.1 Obsługa mapy ... 74
7.3.4 Patronus Child ... 74
7.3.4.1 Usługi działające w tle ... 75
8 Testowanie ... 77
8.1 Testowanie serwera i aplikacji Silverlight ... 77
8.1.1 Wyświetlanie map ... 78

8.1.2 Automatyzacja testów ..... 78
8.2 Testy aplikacji mobilnej ..... 79
8.2.1 Użycie emulatora ..... 79
8.2.2 Automatyzacja testów ..... 81
8.2.3 Testy na rzeczywistym urządzeniu ..... 83
9 Podsumowanie ..... 85
10 Bibliografia ..... 86
11 Spis załączników ..... 88
12 Spis ilustracji ..... 88

# 2 Wstęp

Przedstawiona praca inżynierska porusza kilka problemów z dziedziny programowania, głównie aplikacji mobilnych, ale również dziedzin pokrewnych. Głównym jej tematem przedstawienie procesu tworzenia aplikacji na platformę Windows Phone 7.5 Mango. Przykładowym systemem jaki będzie tutaj tworzony jest system monitorowania położenia dzieci o nazwie Patronus.

Specyficzne pojęcia jakie będą pojawiały się w pracy są wyjaśnione w słowniku pojęć (rozdział 3). Właściwy etap projektowania natomiast rozpocznie się w rozdziale 4, gdzie nastąpi przedstawienie, oraz porównanie istniejących rozwiązań służących do monitorowania położenia dzieci, w celu wyłonienia funkcjonalności i rozwiązań technicznych, które staną się bazą dla Patronusa.

W kolejnym rozdziale określone zostanie, jakie funkcje powinien oferować wzorowy system monitoringu, tak by wygodne było jego użytkowanie, a do tego zadowolił nawet użytkowników ze specyficznymi wymaganiami. Następuje też tutaj analiza, jakie technologie najlepiej wykorzystać, tak by transmisja informacji była na tyle płynna by nie powodować irytacji osoby obsługującej urządzenia, a także zminimalizować koszty, jakie będą za nią ponoszone. Rozważania tego rozdziału mają na celu dać podstawy do zdefiniowania założeń projektowych dla systemu, który zgodnie z tytułem mojej pracy pozwoli na monitorowanie położenia dzieci, za pomocą aplikacji mobilnej zainstalowanej na ich telefonach. Gdy uda się zdefiniować główne założenia projektu, oraz wyspecyfikować jego funkcjonalności, w następnym, szóstym rozdziale zatytułowanym Architektura, nastąpi przedstawienie wizji systemu od strony programistyczno-technicznej, mianowicie jak on będzie wyglądał ‘od środka’, co będzie potrzebne do jego pracy, w jaki sposób będzie odbywała się transmisja danych, oraz w jaki sposób dane te będą przechowywane.

Następnym etapem mojej pracy będzie przykładowa implementacja fragmentu systemu, który przy określonych warunkach spełni dostatecznie dużo funkcjonalności zaplanowanych na etapie projektu, by było możliwe jego działanie, oraz przetestowanie w celu potwierdzenia jego działania. To właśnie rozdział 7. Tutaj głównie zostanie przedstawiony wspomniany wcześniej system Windows Phone 7.5, który jest wciąż swego rodzaju nowością na rynku, dlatego też chciałbym się mu przyjrzeć z bliska. System jednak nie będzie ograniczał się wyłącznie do działania na smartfonie, dlatego też w tej fazie opiszę pozostałe narzędzia z którymi będę pracował (głównie firmy Microsoft, w celu łatwiejszej implementacji i współpracy z Windows Phone). Skupić chciałem się w tej części na opisaniu głównych zarysów, założeń i sposobie implementacji poszczególnych części systemu, nie omijając bardziej ciekawych rozwiązań.

Rozdział 8 pracy jest poświęcony testowaniu, które zwykle odbywa się niemal równolegle do implementacji, jednak trwa jeszcze po jej zakończeniu. To na co chciałbym tutaj zwrócić uwagę, to to jakie należy przyjąć sposoby testowania konkretnych komponentów systemu, tak by przypadki testowe mogły wykazać poprawną implementację funkcjonalności. Co więcej, wtedy kiedy to możliwe chciałbym zastosować mechanizmy automatycznego testowania zarówno pojedynczych metod, jak i całych klas. Ważną częścią tego rozdziału będzie też użycie emulatora telefonu z systemem Windows Phone, i użycie go do testów przed wdrożeniem i przetestowaniem systemu na rzeczywistym urządzeniu.

Całość pracy kończy krótkim podsumowaniem zawierającym moje spostrzeżenia, które na pewno się nasuną podczas całego procesu tworzenia procesu tworzenia systemu.

# 3 Słownik pojęć

Celem słownika pojęć, jest umożliwienie łatwiejszego zrozumienia pracy.

***System Patronus*** – wszystkie projektowane części, które jako całość pozwalają na pełny monitoring położenia dzieci, razem ze wszystkimi dodatkowymi funkcjonalnościami.

***Patronus Serwer***– centralna część systemu, która zbiera wszystkie informacje od telefonów i przeglądarek, pozwala nimi zarządzać i umożliwia m.in. tworzenie planu dnia dzieci, czy tez przeglądanie jak dany dzień dziecka przebiegał.

***Region*** - wyznaczana na mapie z prostych kształtów, płaska figura, której zadaniem jest określanie granic, w jakich dziecko może się poruszać.

***Obszar z czasem***– jest to region wraz z informacją ile czasu dziecko będzie się w nim znajdować.

***Harmonogram*** – plan dnia dziecka, z wyznaczonymi obszarami, w jakich dziecko powinno się znajdować w określonych godzinach.

***Wpis harmonogramu*** – pojedynczy wpis w harmonogramie, zawierający informacje, o której godzinie i na jakim obszarze będzie przebywało dziecko, oraz co jaki interwał czasu to zaplanowane zdarzenia ma się powtarzać.

***Partonus Parent – aplikacja rodzica)*** – instalowana na urządzeniu mobilnym (smartfon) rodzica aplikacja, która umożliwia monitorowanie położenia dzieci, oraz sprawdzanie jego pozycji na mapie.

***Patronus Child – aplikacja dziecka*** – instalowana na telefonie dziecka aplikacja, której zadaniem jest monitorowanie położenia, oraz sprawdzanie jego zgodności z harmonogramem, a także alarmowanie rodziców o niezgodnościach

***Alarm*** – wiadomość wysyłana przez aplikację dziecka do aplikacji rodzica, w przypadku opuszczenia obszaru, określonego w harmonogramie na daną godzinę, sygnalizująca sytuację gdy telefon wykryje zbyt szybki ruch (np. upadek).

***Patronus Viewer*** – udostępniana przez przeglądarkę internetową aplikacja, która umożliwia konfigurację systemu i zarządzanie użytkownikami i harmonogramami. Nazywana również aplikacją internetową lub zarządzającą.

***Użytkownik*** – osoba używająca systemu Patronus. Rozróżniane są 2 typy użytkowników, rodzice i dzieci, w zależności od tego czy w systemie ich zadaniem jest monitorowanie położenia, czy też są monitorowani.

***Synchronizacja*** – proces wymiany danych pomiędzy telefonami a serwerem, którego celem jest sprawdzenie czy telefony posiadają aktualne dane, i jeśli nie to ich zaktualizowanie.

# 4 Przegląd istniejących rozwiązań

Obecnie oferowanych jest już kilka rozwiązań, które pozwalają w mniejszym lub większym stopniu, oraz przy mniejszym, lub większym nakładzie pracy i pieniędzy monitorować położenie osób posiadających przy sobie urządzenie mobilne. Przeglądając te rozwiązania wybrałem kilka, które chciałbym tutaj przedstawić, pokazując, na jakiej zasadzie działają, co dokładnie oferują, oraz jak je obsługiwać.

Przedstawione poniżej cztery różne systemy monitorowania położenia częściowo wykonuje te same zadania. Jednak różnią się one od siebie zastosowaną technologią, podejściem do problemu, dodatkowymi funkcjonalnościami, czy też ceną, albo po prostu stopniem ukończenia usługi i możliwością zastosowania jej w praktyce.

## 4.1 NaviTracker

NaviTracker [1] jest systemem którego założenia są dosyć proste. Po pierwsze należy zarejestrować się na witrynie systemu, wybierając swoją nazwę użytkownika (ang. Login) i hasło. Dalej wymagane jest posiadanie telefonu obsługującego aplikacje JAVA, na którym zainstalujemy aplikację umożliwiającą śledzenie. Producent zaleca, żeby domyślnie ustawić łączenie się z Internetem poprzez GPRS, gdyż, kiedy ustawimy, jako domyślną sieć WiFi, to nie wszędzie będzie możliwe wysłanie położenia na serwer. NaviTracker można pochwalić za prostotę obsługi, ponieważ aplikacja na telefonie ma w zasadzie trzy pozycje: rozpoczęcie śledzenia, wpisanie w ustawieniach naszego loginu i hasła, oraz opcję wyjścia. Po takiej konfiguracji telefonu możemy przejść do testowania jak system sprawuje się w praktyce.

System ten jednak nie został ukończony w czasie pisania pracy. Logowanie na stronie internetowej jest sztuczne, aplikacja na telefon nie chce działać w tle, oraz nie ma żadnego kontaktu z twórcami NaviTracker.. Opisuje tutaj ten system jedynie ze względu na prostą ideę, i wygodne zastosowanie.

## 4.2 Navitracer

Druga usługa nazywa się bardzo podobnie to opisywanej wcześniej. Producentem urządzeń Navitracer [2] jest niemiecka firma **Foltyn Industriesystemelektronik GmbH.** Rozwiązanie to opiera się na zakupie jednego z trzech rodzajów urządzeń (w wersji Ligth, zwykłej i Pro dla firm), oraz karty SIM, która jest w nich umieszczana, a także wymagane jest miesięczne płacenie abonamentu.

Gdy już wszystkie te punkty zostaną spełnione, i mamy przykładowo przy sobie małe urządzenie Navitracer Light, przypominające wielkością telefon komórkowy, możemy go użyć w bardzo prosty sposób. Po uruchomieniu wystarczy je mieć po prostu przy sobie, i urządzenie samo zajmie się przesyłaniem informacji o pozycji za pomocą sieci GSM i transmisji GPRS do serwera Navitracer.

Dalej już, wystarczy się nam zalogować na stronie, bez dodatkowego oprogramowania. Jednak by w pełni zobaczyć funkcjonalność Navitracer'a musimy wprowadzić do swojego konta urządzenie, używając do tego jego numeru IMEI, oraz numeru autoryzacyjnego, który znajduje się na opakowaniu.

Producent wymienia, że jego produkt może znaleźć zastosowanie przy zapewnieniu bezpieczeństwa dzieciom, osobom starszym lub chorym, a także nawet zwierzętom domowym,

przypinając im zakupione urządzenie do obroży. Oprócz tego z dodatkowych możliwości można tutaj wymienić zastosowanie przy monitoringu floty samochodowej, z takimi funkcjonalnościami jak nadzór prędkości samochodu, alarmowanie w przypadku wyjechania poza dozwoloną trasę przez kierowcę, czy nawet kontrola zużycia paliwa.


**Rysunek 1. Strona główna Navitracer**

Ostatnim z aspektów urządzeń Navitracer, jest możliwość jego użycia np. po podróży w portalach społecznościowych, mianowicie podzielenie się z innymi trasami, które zostały zapisane na serwerze w trakcie trwania wyjazdu. Co więcej, także i tutaj mając przy sobie urządzenie można czuć się bezpieczniej, gdyż nawet w przypadku zagubienia czy wypadku łatwo będzie nas odnaleźć.

Wracając jednak do aspektu technicznego, przyjrzyjmy się wspomnianej metodzie przesyłania danych lokalizacyjnych. Urządzenie mobilne jest wyposażone w moduł GPS, który pozwala na określenie pozycji, a także korzysta z sieci GSM do komunikacji z głównym serwerem. Twórcy postawili na prostotę obsługi, zatem użytkownik w zasadzie nie musi robić nic poza posiadaniem urządzenia (przynajmniej w wersji Light, w innych jest możliwość ich dodatkowego spersonalizowania i skonfigurowania na potrzeby klienta). Mając już dane na temat położenia, tak jak już było wspomniane, za pomocą transmisji GPRS dane te przesyłane są na serwer. Pozwala nam na to comiesięczne opłacanie abonamentu. Natomiast, gdy aktualnie urządzenie nie znajduje się w zasięgu sieci, informacje na temat pozycji są buforowane w urządzeniu i wysyłane przy ponownym nawiązaniu połączenia.

## 4.3 Gdzie Jest Dziecko

Usługa lokalizacyjna o nazwie Gdzie Jest Dziecko [3], jak sama nazwa wskazuje, skupia się na monitorowaniu położenia dzieci przez rodziców. Twórcy na swojej stronie przygotowali interaktywną prezentację, która naświetla ideę działania całego systemu monitorowania.

Rozwiązanie przez nich zastosowane, opiera się na komunikacji za pomocą wiadomości tekstowych (SMS), które też służą do dokonywania płatności za usługę lokalizowania.

Możliwości monitorowania swoich dzieci mamy tutaj dwie, za pomocą przeglądarki internetowej, bądź instalując wersję na urządzenia mobilne oprogramowania Gdzie Jest Dziecko. Natomiast, jeśli chodzi o urządzenia monitorowane, to nie jest na nich instalowana żadna aplikacja. Konieczne jest jednak aktywowanie najpierw numeru rodzica, który będzie nadzorował dziecko, a następnie aktywowanie usługi na telefonie dziecka. Wszystko to jest przejrzyście opisane we wspomnianej prezentacji, i polega na wysłaniu kilku wiadomości tekstowych. Kolejnym krokiem jest założenie konta umożliwiającego monitorowanie w usłudze internetowej. Stąd też mamy dostęp do takich opcji jak dodawanie monitorowanego telefonu dziecka, czy definiowanie miejsc, w których dzieci mogą przebywać.


**Rysunek 2. Strona główna usługi lokalizacyjnej Gdzie Jest Dziecko**

Cały mechanizm jest obecnie dostępny dla numerów najpopularniejszych sieci komórkowych w kraju. Ponieważ to właśnie poprzez operatora położenie telefonów, a właściwie kart SIM jest namierzane. Wprowadzony tutaj został jeszcze system punktów, które można kupować płacąc wysłaniem wiadomości SMS. Każdy kupiony punkt można wykorzystać do jednorazowego zlokalizowania aktualnego położenia danej osoby. Samo lokalizowanie jest dostępne nie tylko przez Internet, ale również w każdym momencie w telefonie komórkowym, pod warunkiem wysłania wiadomości z zapytaniem o położenie naszej pociechy. Wiadomość ta ma określony schemat, po czym w SMS-ie zwrotnym dostaniemy opis położenia dziecka, (jeśli znajduje się ono w którymś ze zdefiniowanych przez nas obszarów), bądź też jego współrzędne geograficzne. W przypadku użycia do tego przeglądarki internetowej, pozycja zostaje zaznaczona na mapie.

Interesującym aspektem, który już został wspomniany jest też możliwość z korzystania z usługi lokalizacyjnej za pomocą aplikacji na urządzeniu mobilnym. Obecnie można z niej korzystać na smartfonach z systemami Android i Windows Mobile, lub tez pozwalających na uruchamianie aplikacji **JAVA** w wersji MIDP 2.0.

Jak to zwykle bywa przy tego typu usługach, istnieje również wersja Premium, w której to m.in. mamy możliwość otrzymywania powiadomień, czy nasze dziecko znajduje się o określonej godzinie w podanej przez nas okolicy, przykładowo w szkole, a także oferowane jest zapamiętywanie ostatniej pozycji gdzie było dziecko przed wyłączeniem telefonu, lub utratą zasięgu.

Wracając jednak do metody określania pozycji, to odbywa się ona za pośrednictwem operatora sieci komórkowej, czyli za pomocą technologii GSM, dzięki czemu niewymagane jest posiadanie w telefonie dziecka modułu GPS.

Usługa Gdzie Jest Dziecko jest bardzo prosta w użyciu, nie wymaga dokupowania specjalnych urządzeń, czy wykupywania abonamentu, a koszty wiadomości SMS, za pomocą, których odbywają się wszystkie opłaty, nie należą do wygórowanych. Jednak z drugiej strony, w podstawowej wersji nie jest prowadzony monitoring w czasie rzeczywistym, skąd może wyniknąć pewne opóźnienie, zanim rodzic dowie się o zmianie położenia swojej pociechy.

## 4.4 SpySat

Kolejną z przedstawianych usług lokalizacyjnych jest SpySat [4]. Z punktu widzenia programistów aplikacji mobilnych rozwiązanie najciekawsze, ponieważ pozwala na korzystanie z udostępnionego przez twórców protokołu komunikacyjnego. Sam protokół opiera się na protokole http i okresowym wywołaniu określonego adresu URL (Rysunek 4). Uściślić trzeba, że służy on jedynie do przesłania na serwer SpySat informacji o naszym płożeniu, zatem umożliwia nam napisanie własnej aplikacji mobilnej wykorzystującej GPS.Oczywiście jest też drugie rozwiązanie, w którym nie musimy tego robić. Możemy skorzystać z gotowej aplikacji, która jest dedykowana dla urządzeń z systemami Android lub Windows Mobile.

Aktualnie SpySat dostarcza oprogramowania na urządzenia mobilne z systemami Windows Mobile oraz Google Android, jednak otwartość protokołu przesyłania danych powoduje, że możliwe jest stworzenie oprogramowania na dowolną platformę sprzętowo-programową.

**Opis protokołu komunikacyjnego spysat**

Protokół komunikacyjny opiera się na HTTP i polega na okresowym wywoływaniu ( zalecany okres czasu to 15 sekund lub jego wielokrotność) specjalnego URL ze zmiennymi w GET w następującym formacie:

`http://spysat.pl/dh/g.php?u=[LOGIN]&p=[PIN]&x=[LAT]&y=[LON]&z=[ALT]&s=[SPEED]`

**Opis parametrów:**

- **[LOGIN]** - login użytkownika w spysat
- **[PIN]** - numeryczny PIN urządzenia który musi być taki sam jak ten który podaliśmy przy definiowaniu urządzenia
- **[LAT]** - szerokość geograficzna w formacie kątowym (bez znaków minut i stopni) np. 50.5730831623077 - separator dziesiętny - kropka
- **[LON]** - długość geograficzna w formacie kątowym (bez znaków minut i stopni) np. 23.0535784959793 - separator dziesiętny - kropka
- **[ALT]** - wysokość nad poziomem morza - wyrażona w metrach, podawana jako liczba całkowita (części dziesiętne są pomijane)
- **[SPEED]** - prędkość poruszania się - - wyrażona w kilometrach na godzinę, podawana jako liczba całkowita (części dziesiętne są pomijane)

Można zatem tworzyć własne programy na dowolne urządzenie mobilne które posiada techniczne możliwości odbierania danych GPS i wysłania ich za pośrednictwem Internetu do naszego serwera

**Rysunek 3. Opis protokołu używanego do komunikacji z serwerami SpySat**

W związku z tym, na jakie platformy przeznaczone są aplikacje monitorowane, wybór urządzeń jest dużo mniejszy niż chociażby w opisywanej wcześniej usłudze Gdzie Jest Dziecko.


Rysunek 4. Schemat działania systemu SpySat

Ideę działania technologii SpySat pokazuje schemat na Rysunku 5. Osoba posiadająca na swoim urządzeniu mobilnym dostęp do Internetu przesyła za pomocą aplikacji cyklicznie swoje położenie. Dane te możemy następnie zobaczyć logując się na swoje konto w systemie. Konto to musimy założyć przed rozpoczęciem monitorowania, gdyż przesyłanie informacji na serwer wymaga podania w wołanym przez urządzenie mobilne adresie URL nie tylko pozycji, ale także danych identyfikujących, komu te informacje potem będą przestawione. Ostatnim faktem, który jest dużą zaletą SpySat, jest to, że usługa jest w pełni darmowa, oprócz oczywiście kosztów związanych z przesyłaniem danych do systemu.

## 4.5 YourCargo

Co prawda YourCargo [5] przeznaczone jest głównie do namierzania pozycji pojazdów, w celu nadzorowania przewozu towarów, oraz kontrolowania czasu dostaw. Udostępniany jest bezpłatny system GPS, który w czasie rzeczywistym sprawdza pozycję kierowcy wyposażonego w telefon z system operacyjnym Symbian. Żeby zacząć użytkować system należy na telefonie zainstalować oprogramowanie od producenta, natomiast sam monitoring, tak jak w przypadku poprzednich usług odbywa się za pośrednictwem przeglądarki internetowej.

Funkcjonalności, które wyróżniają YourCargo to chociażby udostępnienie na stronie internetowej bramki sms, ułatwiającej komunikację z kierowcami (jakkolwiek odpowiadanie na wiadomości tekstowe przez kierowców w czasie jazdy nie jest godne pochwały). Z ciekawych opcji z poziomu przeglądarki mamy możliwość przeglądania pojazdów i ich ładunków, oraz planowania najkrótszych tras, a także szacowania kosztów przejazdu.

W zasadzie usługa, jako swój główny atut podaje udostępnienie giełdy transportowej. Umożliwia to ogłaszanie się przewoźników i bezpośredni z nimi kontakt. Dzięki YourCargo, po zalogowaniu się na konto na mapie możemy wyszukać gdzie najbliżej nas jest gotowy do wyjazdu samochód ciężarowy, a także pominąć w kontakcie z przewoźnikiem pośredników, a tym samym zmniejszyć koszty transportu.

Co prawda przeznaczenie systemu jest inne niż temat mojej pracy, ale uznałem go za warty uwagi w dziedzinie dostępnych na rynku rozwiązań lokalizujących.

## 4.6 Usługi oferowane przez operatorów sieci komórkowych

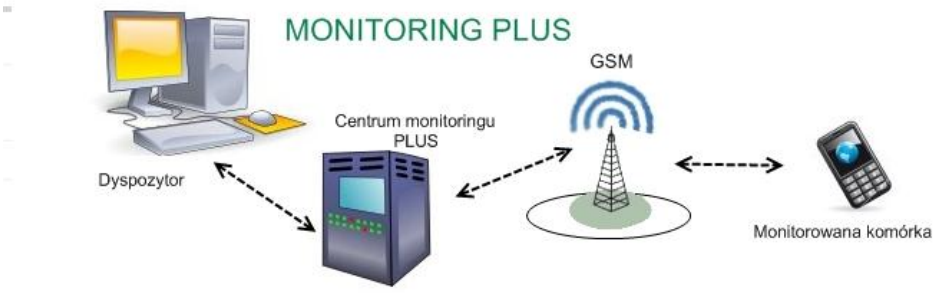
Podrozdział ten w celu pełnego przeglądu rozwiązań prezentuje w zasadzie niemal identyczne usługi monitoringu położenia urządzeń wyposażonych w kartę SIM bezpośrednio przez operatora sieci. Prezentując pierwszą z nich skupiłem się na pokazaniu wszystkich jej funkcjonalności, natomiast przy kolejnych chciałem zwrócić uwagę głównie na różnice pomiędzy kolejnymi rozwiązaniami.

***Monitoring SIM*** [6] oferowane przez T-Mobile, ***Nawigator*** [7] za który odpowiada Orange, oraz ***Monitoring Plus*** oferowane przez Plus są to wspomniane usługi nawigacyjne najpopularniejszych operatorów sieci komórkowych w Polsce.

Co prawda usługi te różnią się nieco w zakresie oferowanych funkcjonalności, ale główne ich funkcje pozostają wspólne. Mianowicie chodzi tutaj o lokalizowanie telefonów komórkowych i możliwość późniejszej obserwacji, gdzie użytkownik przemieszczał się z telefonem w ciągu dnia.

Jako ciekawostkę warto tutaj przytoczyć, że operator sieci Orange skrótowo opisuje metodę, jaka jest wykorzystywana do lokalizowania terminalu z kartą SIM. Mianowicie chodzi tutaj o LBS (Location Based Service). Dzięki temu możliwe jest sprawdzenie pozycji telefonu który nie posiada modułu GPS. Jest jednak to obarczone błędami przybliżeń, gdyż wtedy szacowana jest odległość od najbliższych nadajników sieci GSM, i na tej podstawie wyznaczana jest lokalizacja.

Dodatkowo operatorzy udostępniają również dla telefonów obsługujących technologię JAVA różnego rodzaju aplikacje, pozwalające na bieżąco sprawdzać położenie innych telefonów dla których usługa została aktywowana.




**Rysunek 5. Monitoring Plus - schemat działania**

## 4.7 Porównanie

Nadszedł czas na zestawienie wszystkich zalet i wad powyższych projektów. Jak można było zauważyć różnią się one zarówno sposobem rozpoczęcia pracy, wymaganymi do niej urządzeniami, kosztami, możliwością nadzorowania pozycji w czasie rzeczywistym, czy też dodatkowymi funkcjonalnościami. Usługi oferowane przez operatorów sieci komórkowych są uwzględnione w zestawieniu wspólnie, gdyż ich działanie jest niemal analogiczne.

Przyglądając się zestawionym w tabeli właściwościom poszczególnych systemów możemy sami zdecydować, który z nich najlepiej spełnia nasze oczekiwania, czy to pod względem prostoty obsługi, minimalizacji kosztów, czy też posiadania konkretnego urządzenia.

W praktyce jedynym systemem, którego nie mogę polecić jest Navitracker, z uwagi na fakt, że projekt ten nie jest skończony.

Dla rodziców, którzy nie chcą inwestować w drogie urządzenia i chcą skorzystać z posiadanych telefonów komórkowych najlepiej nada się Gdzie Jest Dziecko. Aczkolwiek mimo nazwy usługa może być równie dobrze użyta do lokalizowania nie tylko dzieci.

W przypadku, gdy zależy nam na prostocie i niezawodności, oraz możemy wydać nieco więcej pieniędzy, rozwiązaniem najlepszym okazuje się Navitacer, który udostępnia własne gotowe urządzenia gotowe do działania.

Jeśli natomiast chcemy mieć większą kontrolę nad tym jak i kiedy informacje o naszej lokalizacji będą przesyłane, naprzeciw wychodzi nam usługa SpySat.

Widzimy jednak, że część z powyższych usług jest niekompletna, część jest ciągle rozwijana, prawdopodobnie też każdy mógłby zauważyć, że przydatne byłoby dodatnie do nich innych jeszcze funkcji, albo, że niektóre z oferowanych są zbędne. Idąc tym tropem i analizując funkcjonalności prezentowanych systemów, oraz sposób ich wykonania w następnych rozdziałach postaram się naświetlić, które z nich, według mnie, są warte uwagi i jakie inne wymagania powinny być spełnione przez system monitorowania, żeby można go było nazwać dobrym. Co więcej ważnym aspektem monitorowania jest rozważenie użytej do tego celu technologii, a także zanalizowanie kosztów, jakie będziemy wtedy ponosić. Stąd kolejny etap pracy będzie to projekt systemu monitorującego położenie dziecka w ciągu dnia, na który będą się składać funkcjonalności wyłonione podczas tej analizy jako najbardziej praktyczne

System o nazwie Patronus, który chciałbym zaprojektować i zaimplementować został również uwzględniony w powyższej tabeli, aby była możliwość porównania go z pozostałymi systemami. Starałem się żeby z założenia był to system jak najbardziej przyjazny rodzicom i to właśnie dla nich jest dedykowany, stąd chociażby taka jego funkcjonalność jak tworzenie planów dnia dla dzieci. Docelowo Patronus ma działać na dowolnej platformie mobilnej, jednak w tej pracy skupiam się na Windows Phone. Do zarządzania systemem najwygodniejsze będzie użycie przeglądarki internetowej co oszczędzi użytkownikom instalowania dodatkowego oprogramowania na komputerze i pozwoli na dostęp do niego z dowolnego miejsca z połączeniem do Internetu. To natomiast co wyróżnia Patronusa spośród pozostałych systemów to elastyczność w jego konfiguracji, oraz prostota w planowaniu zajęć dla dzieci z określaniem obszarów gdzie te zajęcia mają się odbywać. Ponadto umożliwia on otrzymywanie alarmów o sytuacjach gdy dziecko nie będzie w miejscu które przewiduje harmonogram.

**Tabela 1. Porównanie usług monitorujących położenie.**

| ***Usługa*** | **Navitracker** | **Navitracer** | **Gdzie Jest Dziecko** | **SpySat** | **YouCargo** | **Usługi operatorów** | **Patronus – planowany system** |
|---|---|---|---|---|---|---|---|
| *Wymagane urządzenia* | Telefon +JAVA | Urządzenie producenta | Telefon komórkowy | Windows Mobile, Android. | Symbian | Telefon + JAVA | Windows Phone – w tej wersji, Projekt dla dowolnego telefonu |
| *Metoda prezentacji położenia* | Przeglądarka internetowa | Przeglądarka internetowa | Przeglądarka internetowa, SMS, urządzenie mobilne | Przeglądarka internetowa | Przeglądarka internetowa | Przeglądarka internetowa, Telefon + Java | Przeglądarka internetowa, Smartfon |
| *Metoda przesyłania danych* | GPRS | GPRS | SMS | Specjalnym protokołem przez http | Brak danych | SMS, GPRS | Internet, SMS |
| *Określanie pozycji* | GPS/Internet | GPS | GSM | GPS | GPS | GSM | GPS |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *Koszty* | Darmowa | 40 zł/ miesięcznie – abonament | Płatne SMS'em | Darmowa | Darmowa | Płatna | Darmowa |
| *Śledzenie w czasie rzeczywistym* | Tak | Tak | Tylko w wersji Premium. | Tak | Tak | Tak | Częstotliwość ustalana przez użytkownika |
| *Działanie przy braku zasięgu* | Brak działania | Buforowanie danych | Brak działania | Dane nie są przesyłane | Brak danych | Brak danych | Buforowanie danych |
| *Stopień ukończeniu usługi* | Nie działa | Ciągle rozwijana | Ciągle rozwijana | Ukończona | Ukończona | Ukończona | Projektowana |
| *Definiowanie harmonogramu dnia* | Nie | Nie | W wersji premium | Nie | Nie | Nie | Tak |
| *Tworzenie obszarów* | Nie | Tak | W wersji premium | Nie | Nie | Nie | Tak |
| *Pytanie o pozycję z telefonu* | Nie | Nie | Tak | Nie | Nie | Tak | Tak |
| *Alarmowanie w określonych sytuacjach* | Nie | Tak | W wersji premium | Nie | Nie | Nie | Tak |
| *Konieczny system centralny* | Tak | Tak | Tak | Tak | Tak | Tak | Tak |

# 5 Analiza

Rozpoczynając etap projektowania, zacznę od definiowania wymagań, jakie możemy postawić systemowi monitorującemu położenie dzieci. Podczas procesu projektowania starałem się zachować zgodność z zasadami inżynierii oprogramowania [8]. Możemy stwierdzić, że jest to bardzo szeroki temat, a do tego płynny. Jest tak, ponieważ wymagania te będą się różnić na bardzo wiele sposobów, w zależności od tego jak rodzice wyobrażają sobie całe to przedsięwzięcie. Nie mamy sprecyzowanego konkretnego odbiorcy, a jedynie grupę odbiorców, czyli rodziców. Trudno, zatem będzie o spełnienie wymagań każdego z nich, jednak można określić uniwersalne funkcje, jakie powinny być możliwe do użycia. Wymagania jednak, nie dotyczą jedynie tego, co cały system pozwala zrobić, ale również mają wpływ na aspekt techniczny, czyli jak to ma być wykonane. Wiąże się to z tym, że np. od użytego sposobu przesyłania danych będzie zależało to jak kosztowne będzie korzystanie z monitorowania (aczkolwiek te zagadnienia będą rozważane w następnym etapie, podczas określania architektury systemu).

Do pozostałych kwestii które należy jeszcze rozważyć należą:

- Określenie typu używanego urządzenia
- Monitorowanie w czasie rzeczywistym, cyklicznie, bądź zależnie od konfiguracji
- Metoda powiadamiania rodzica (natychmiastowa wiadomość SMS, na żądanie, czy jedynie przez internet)

Wszystkie te pytania mają na celu zmusić do zastanowienia się, co tak naprawdę jest konieczne do przygotowania w projekcie, a co jest dodatkiem.

## 5.1 Analiza dziedziny

Zacznijmy od tego, co chcielibyśmy dokładnie móc robić w systemie monitorującym? Żeby to określić, musimy pomyśleć, co byłoby najwygodniejsze dla przeciętnego rodzica w Polsce. Zatem, zapewnie ważną rzeczą jest oczywiście możliwość sprawdzenia gdzie obecnie jest jego dziecko, tylko, pozostaje kwestia jak to zrobić? Możemy do tego użyć programu zainstalowanego na komputerze, bądź przeglądarkę internetową. Wtedy jednak potrzebujemy mieć przy sobie komputer z dostępem do Internetu, a jeśli jesteśmy w podróży to ta możliwość odpada. W takim razie zakładając, że rodzicowi zależy żeby w każdym momencie w domu i poza nim móc zobaczyć gdzie jest jego dziecko, najlepszym rozwiązaniem będzie wykorzystanie telefonu komórkowego, bądź też smartfonu, na którym można zastosować wygodniejszy interfejs użytkownika.

Innym aspektem, który należy rozważyć, jest to czy chcemy być powiadamiani, gdy dziecko znajdzie się nie tam gdzie powinno, np. nie przyszło do szkoły. Praca i inne zajęcia w ciągu dnia nie pozwolą, co chwila zerkać do telefonu w celu sprawdzenia położenia dziecka, poza tym byłoby to bardzo irytujące, oraz w końcu zaniechalibyśmy systematycznego sprawdzania, co się dzieje z naszą pociechą. Dlatego drugą z głównych funkcjonalności powinno być powiadamianie nas o sytuacjach awaryjnych, spóźnieniach do szkoły, jechaniu w przeciwnym kierunku w przypadku pomylenia autobusu itp. Jednak w jakiś sposób musimy określić, kiedy i gdzie ma się dziecko znajdować, najlepiej jeszcze żeby dało się wyznaczyć obszar na mapie. Do tego celu wygodniej by było użyć komputera, ze względu na większy ekran i wygodniejszą obsługę.

Zatem na ten moment wiemy, że część czynności będziemy wykonywali na telefonie, a wyznaczanie obszarów gdzie syn lub córka może przebywać będzie potrzebowało komputera. Ogólny zarys tego, co chcemy móc robić w systemie mamy określony. Zastanówmy się teraz, co jeszcze byłoby przydatne? Przykładowo, o jakich sytuacjach chcielibyśmy się natychmiast dowiadywać? Jedną z takich sytuacji są na pewno upadki dziecka, zwłaszcza w zimie ryzyko takie jest większe. Nawet, jeśli upadek jest niegroźny, lepiej być o nim powiadomionym, żeby przynajmniej wiedzieć o takim fakcie i móc odpowiednio zareagować. Telefon, w który wyposażymy dziecko może taką możliwość udostępniać, i w przypadku gwałtownego upadku powiadamiać rodzica.

Przydatną funkcją może być również w przypadku, gdy nie mieliśmy czasu sprawdzać położenia w ciągu dnia, archiwizowanie wszystkich informacji, które potem za pomocą przeglądarki internetowej czy innego programu możemy przejrzeć. Wygodnie również by było, gdyby przebieg aktualnego dnia dzieci był możliwy do obejrzenia na telefonie, które mają obecnie na tyle duże pojemności, że nie stanowi to problemu.

Ponadto, z usług konfigurujących warto mieć możliwość łatwego zmieniania harmonogramu dnia, oraz wysyłania go na telefon monitorowany, żeby na bieżąco miał informacje o tym jak przebiega plan dnia dziecka, a także by wspomniane już dane, które mają być archiwizowane po całym dniu były przesyłane w jedno miejsce, skąd będą już łatwo dostępne.

## 5.2 Identyfikacja aktorów

Aktorzy, jacy występują w systemie nie są do końca tak oczywiści jak mogłoby się wydawać. Na pierwszy rzut oka możemy stwierdzić, że będą to rodzic i dziecko. Oprócz nich dużą role w systemie pełni aplikacja monitorowana, gdyż to do niej należy sprawdzanie położenia z harmonogramem, wysyłanie/odbieranie powiadomień, czy alarmowanie o sytuacjach wyjątkowych. Jeżeli zaś chodzi o aplikację monitorującą, to również zaliczamy ją do grona aktorów, aczkolwiek jej rola sprowadza się do alarmowania rodzica i wyświetlania powiadomień. Wszyscy aktorzy w systemie:

***Rodzic*** – aktor pełniący w systemie znaczącą funkcję. Do jego zadań należy skonfigurowanie urządzeń, utworzenie profili osób w systemie, definiowanie planu dnia dzieci. Jest on powiązany z używaną przez niego aplikacją monitorującą na urządzeniu mobilnym. Rodzicowi są wyświetlane powiadomienia przez wspomnianą aplikację, oraz to on ma możliwość w każdej chwili sprawdzenia położenia swojego dziecka.

***Aplikacja monitorowana (dziecka)*** – głównym zadaniem tego aktora jest cykliczne sprawdzanie położenia, oraz porównywanie go z harmonogramem, a także powiadamianie systemu, lub bezpośrednio rodziców o nietypowych sytuacjach, a także synchronizowanie danych w terminie konfigurowanym przez użytkownika systemu

***Aplikacja monitorująca (rodzica)*** – zadaniem jej jest sygnalizowanie rodzicowi sytuacji alarmowych, m.in. braku odpowiedzi z telefonu dziecka przez określony przedział czasu.

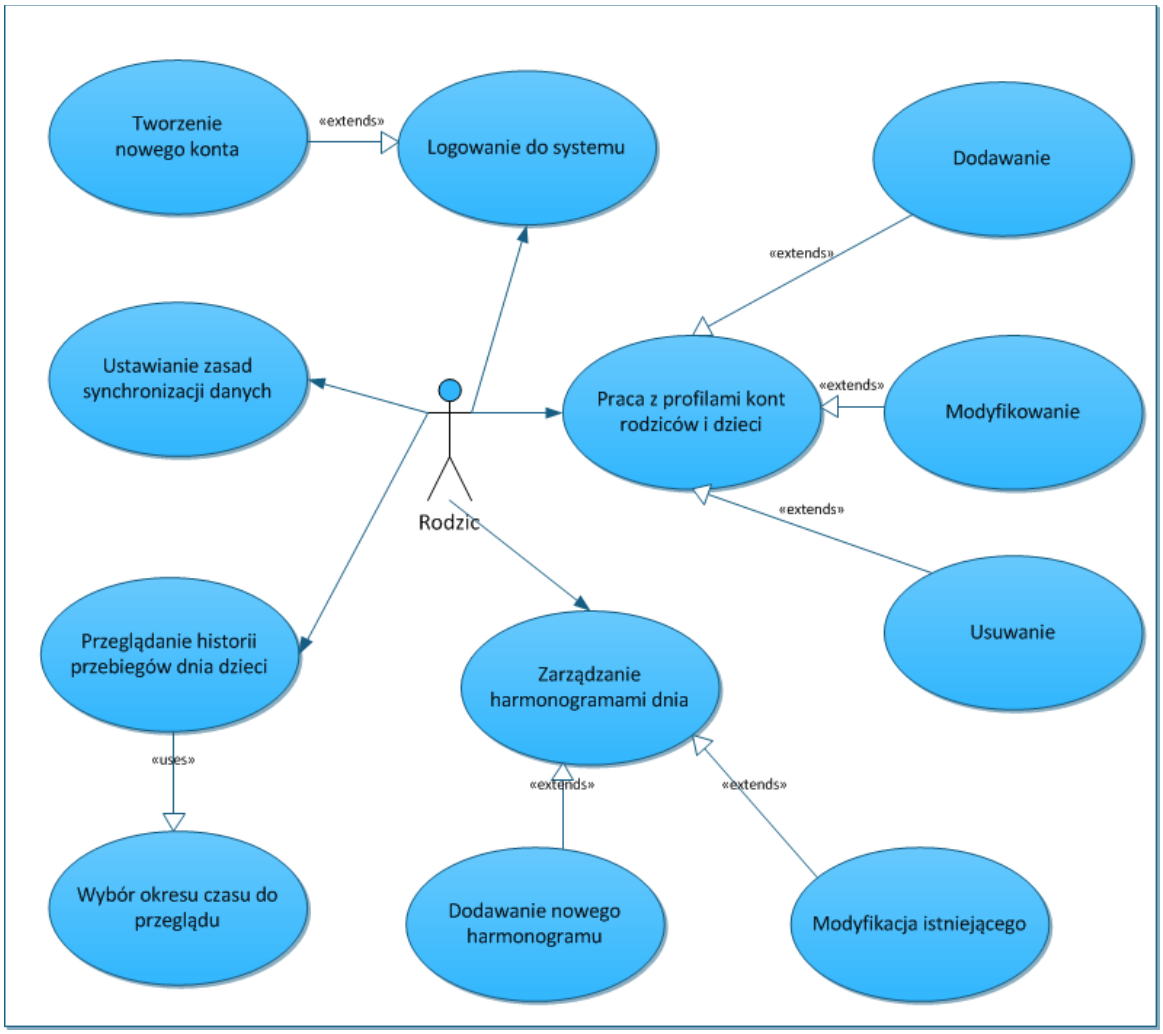
***Dziecko*** – akcje dziecka aktywują do działania aplikację monitorowaną, która po przemieszczeniu się go dokonuje odczytu pozycji. Jednak jeśli chodzi o rolę dziecka jaką ma pełnić w systemie, to będzie to chociażby zgłoszenie alarmu rodzicowi, czy też będzie mogło ono pobrać aktualny plan dnia.

## 5.3 Wymagania funkcjonalne

Wymagania te stanowią usystematyzowany zbiór wszystkich funkcjonalności, które do tej pory były wymieniane, zwłaszcza na etapie analizy. Opisują one wszystkie funkcje, jakie będzie udostępniał system. W kolejnych rozdziałach nastąpi graficzne ich przedstawienie, razem z powiązaniami z aktorami (diagramy przypadków użycia).

### 5.3.1 Konfigurowanie i zarządzanie systemem

Aktorem, którego dotyczy powyższy przypadek użycia jest rodzic. On to będzie logował się w systemie, aby utworzyć profile kont dzieci, by potem możliwa była synchronizacja danych pomiędzy telefonami i serwerem. Także do rodzica należy obowiązek określenia harmonogramów dnia dla każdego dziecka; harmonogramu nie da się ręcznie usuwać, jedynie można go zmodyfikować lub zastąpić nowym. Ustawianie zasad synchronizacji danych polega natomiast na określeniu jak często ma się ona odbywać, oraz jak długo dane mają pozostawać w systemie przed usunięciem.




Rysunek 6. Konfigurowanie i zarządzanie systemem

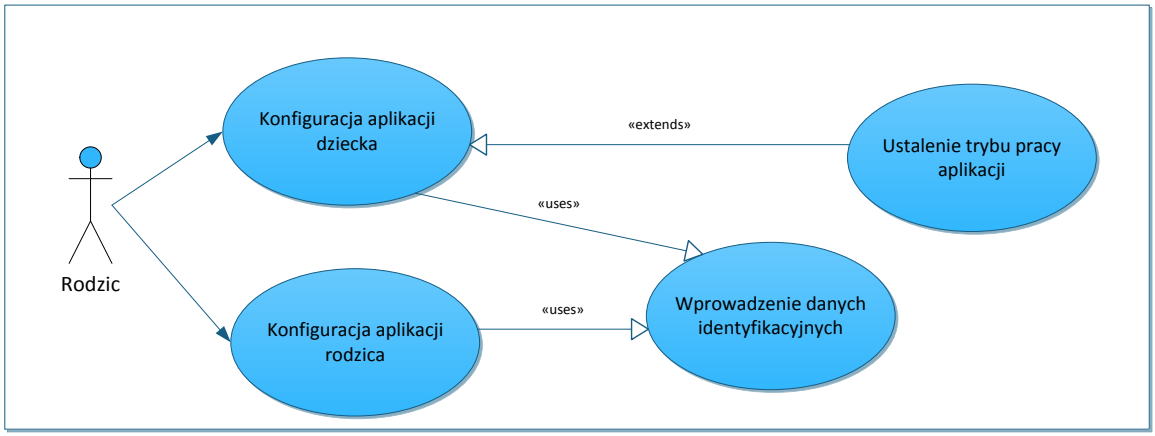
WF1. Konfigurowanie i zarządzanie systemem.
- WF1.1. Logowanie do systemu
  - WF1.1.1. Tworzenie nowego konta.
- WF1.2. Praca z profilami kont rodziców i dzieci.

WF1.2.1. Dodawanie profilu.
WF1.2.2. Modyfikowanie profilu.
WF1.2.3. Usuwanie profilu.
WF1.3. Zarządzanie harmonogramami dnia.
WF1.3.1. Dodawanie nowego harmonogramu.
WF1.3.2. Modyfikacja istniejącego.
WF1.4. Przeglądanie historii przebiegów dnia dzieci.
WF1.4.1. Wybór okresu czasu do przeglądu.
WF1.5. Ustawianie zasad synchronizacji danych.

### 5.3.2 Konfiguracja urządzeń mobilnych

Również w tym przypadku obowiązek dokonania konfiguracji urządzeń spoczywa na rodzicu. Po zainstalowaniu odpowiedniej wersji aplikacji, rodzic musi podać w niej dane identyfikacyjne, takie jak we wcześniej utworzonych w systemie profilach. Dodatkowo w aplikacji dziecka konieczne jest dokonanie synchronizacji danych przy pierwszym jej uruchomieniu, by mogła ona swobodnie rozpocząć pracę (posiadała plan dnia dziecka, oraz nr telefonu rodziców, a także informacje jak mają przebiegać kolejne synchronizacje). Obowiązku takiego nie ma na aplikacji rodzica, aczkolwiek do czasu pierwszej synchronizacji nie będzie ona posiadała informacji o tym, od jakich urządzeń ma spodziewać się przychodzących wiadomości.

WF2. Konfiguracja urządzeń mobilnych
WF2.1. Konfiguracja aplikacji dziecka.
WF2.1.1. Wprowadzenie danych identyfikacyjnych.
WF2.1.2. Ustalenie trybu pracy aplikacji.
WF2.2. Konfiguracja aplikacji rodzica.
WF2.2.1. Wprowadzenie danych identyfikacyjnych.

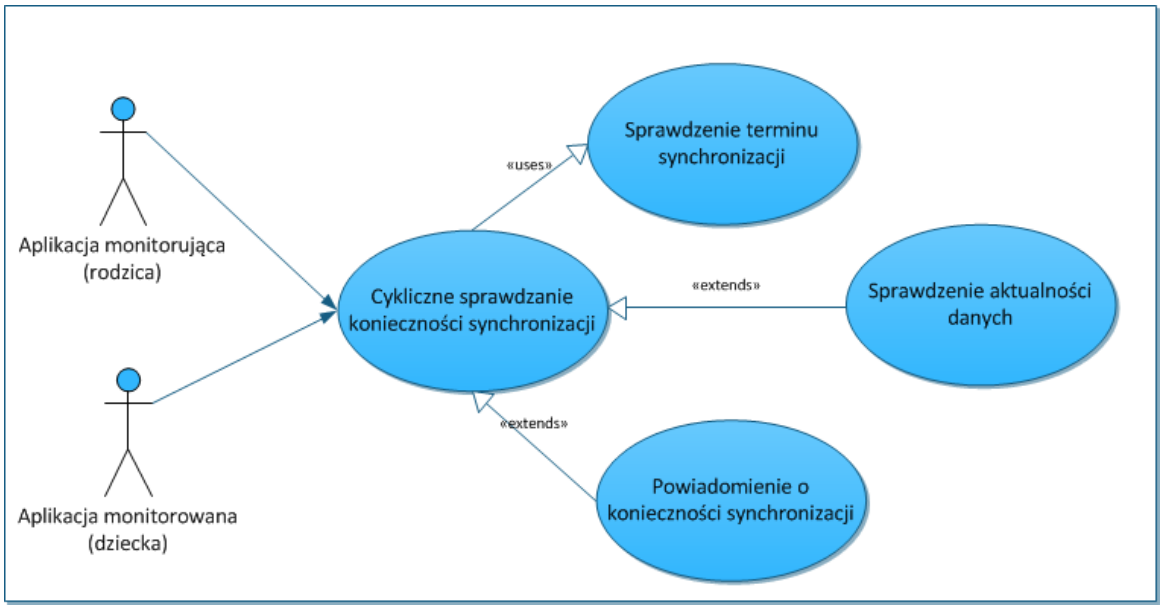



Rysunek 7. Konfiguracja urządzeń mobilnych

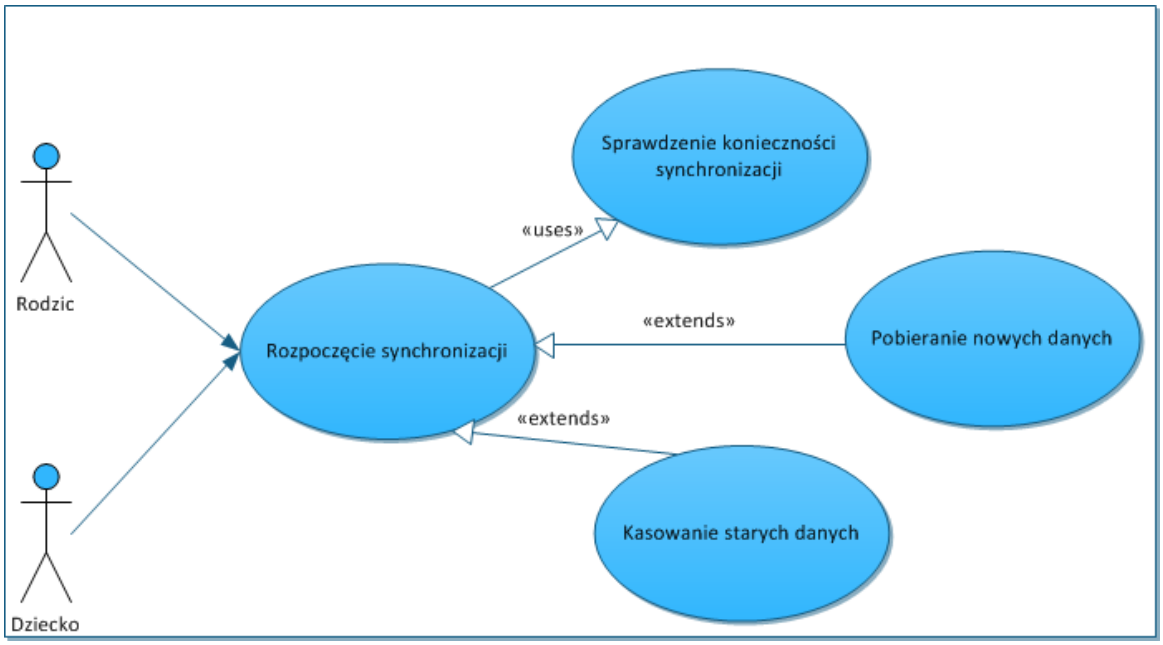
### 5.3.3 Synchronizacja danych

Synchronizacja danych zwykle będzie wykonywana cyklicznie co konfigurowany w systemie interwał czasu. Do aplikacji mobilnych będzie należało zadanie wyświetlenia informacji o konieczności synchronizacji danych, ale to do użytkownika telefonu należy decyzja kiedy ona ma się dokonać. Aplikacja mobilna sprawdzać będzie czy nadszedł już termin synchronizacji danych, i jeśli tak to sprawdzi czy dane po stronie serwera się zmieniły i jeśli tak to wyświetli o tym komunikat.

Podczas aktualizacji danych, smartfon pyta się serwera o konieczność jej dokonania, a następnie oczekuje odpowiedzi. W przypadku gdy dane po stronie urządzenia mobilne wciąż są aktualne, synchronizacja jest zakończona, a jeśli nie, to pobierane są po kolei wszystkie dane z serwera, a dane lokalne są kasowane.




Rysunek 8. Synchronizacja danych - aplikacje mobilne




Rysunek 9 Synchronizacja danych – użytkownicy

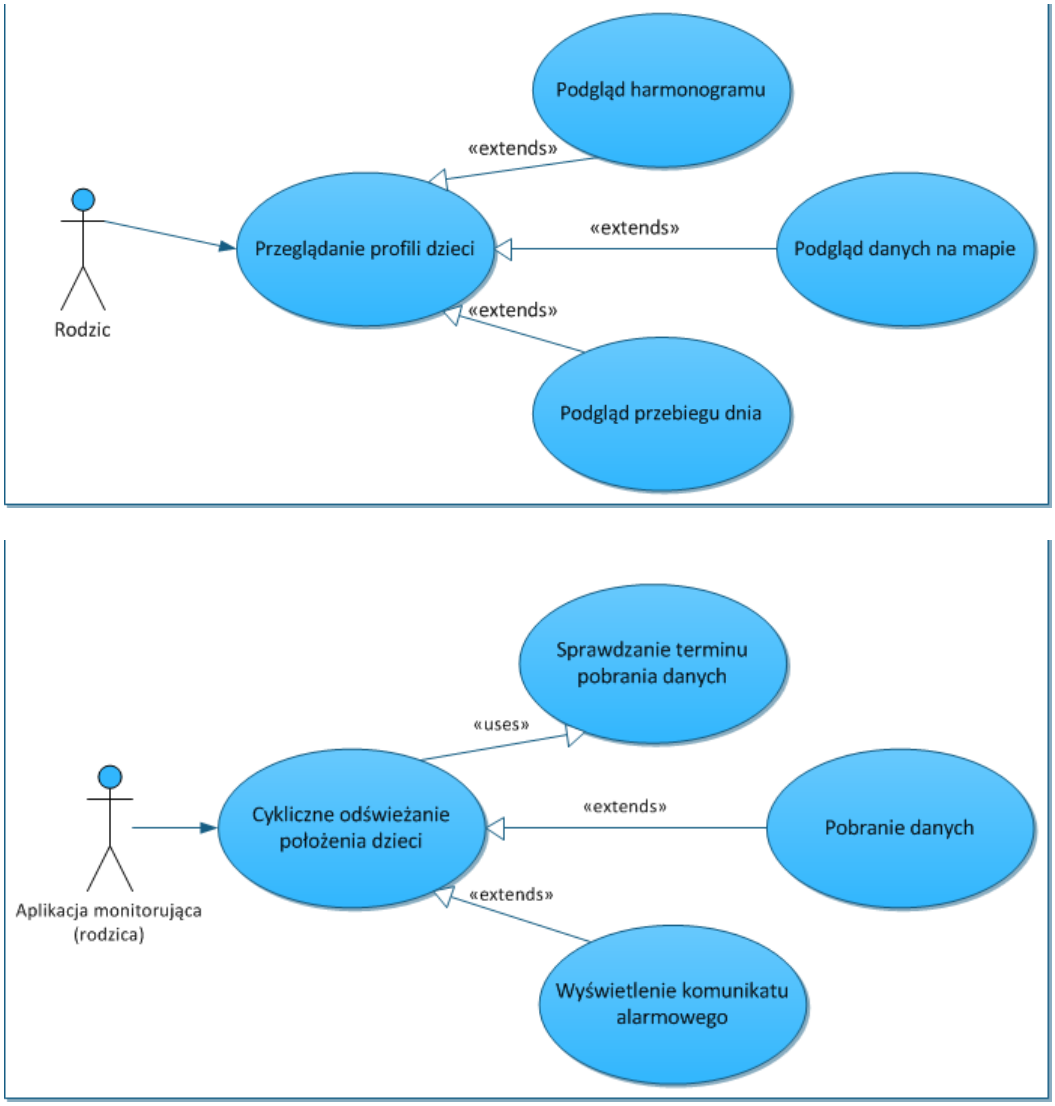
WF1. Synchronizacja danych.
- WF1.1. Cykliczne sprawdzanie konieczności synchronizacji.
  - WF1.1.1. Sprawdzenie terminu synchronizacji.
  - WF1.1.2. Sprawdzenie aktualności danych.
  - WF1.1.3. Powiadomienie o konieczności synchronizacji.
- WF1.2. Rozpoczęcie synchronizacji.
  - WF1.2.1. Sprawdzenie konieczności synchronizacji.

WF1.2.2. Pobieranie nowych danych.
WF1.2.3. Kasowanie starych.

### 5.3.4 Monitorowanie położenia dziecka

Rodzic z poziomu swojego telefonu ma mieć możliwość sprawdzenia gdzie aktualnie znajduje się jego dziecko, a także podejrzeć te informacje na mapie, oraz mieć możliwość sprawdzenia jaki harmonogram dnia jest zaplanowany dla dziecka.

Aplikacja mobilna cyklicznie sprawdza czy pojawiły się nowe informacje o przebiegu dnia dzieci na serwerze i jeśli tak to je pobiera, oraz jeśli istnieje taka konieczność to wyświetla rodzicowi alarm, gdy pobrane dane tego wymagają.



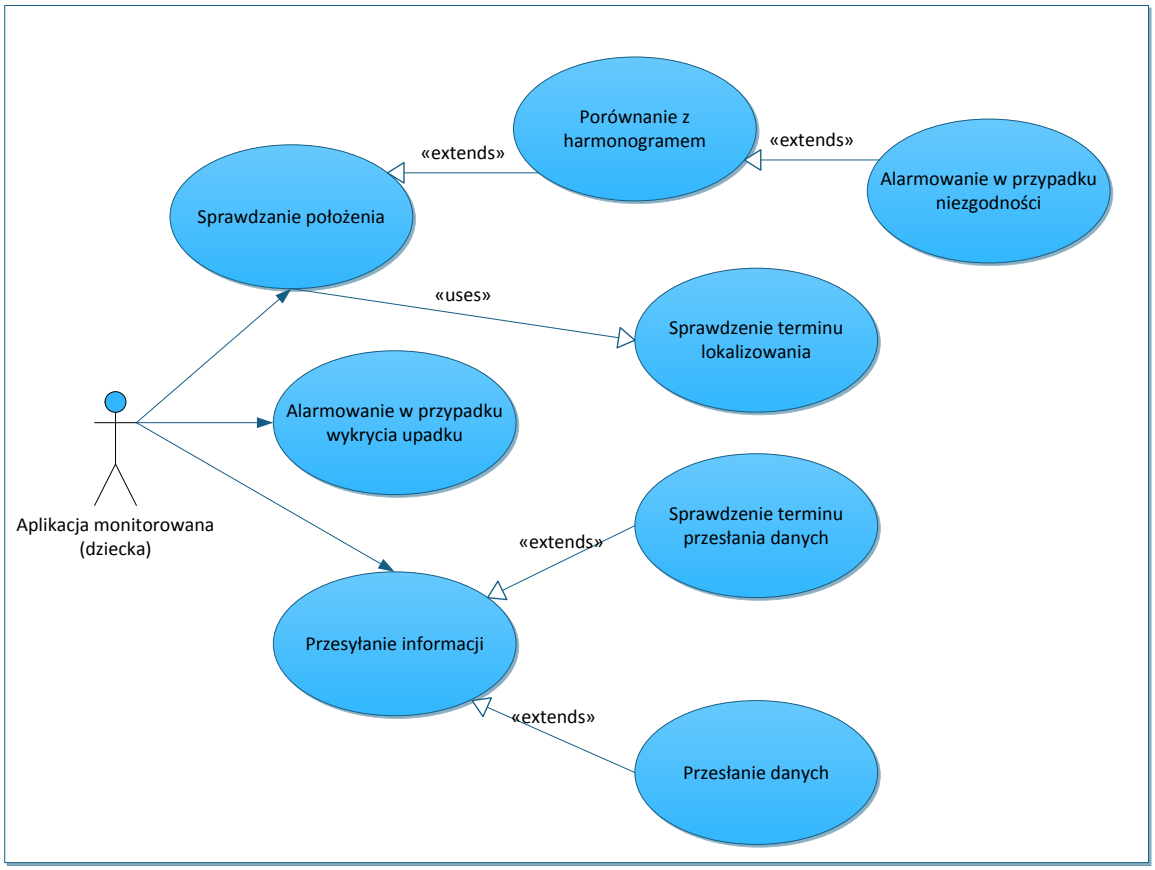

**Rysunek 10. Monitorowanie położenia dziecka z telefonu rodzica.**

WF2. Monitorowanie położenia dziecka z telefonu rodzica.
WF2.1. Przeglądanie profili dzieci.
WF2.1.1. Podgląd harmonogramu.
WF2.1.2. Podgląd przebiegu dnia.
WF2.1.3. Podgląd danych na mapie
WF2.2. Cykliczne odświeżanie położenia dzieci.

WF2.2.1. Sprawdzanie terminu pobrania danych.
WF2.2.2. Pobranie danych.
WF2.2.3. Wyświetlenie komunikatu alarmowego.

Zadaniem aplikacji monitorowanej (dziecka) jest systematyczne sprawdzanie położenia i porównywanie go z harmonogramem. Jeżeli zostanie wykryta niezgodność informacja o tym natychmiast jest wysyłana. W przeciwnym przypadku dane te są buforowane w telefonie i przesyłane co określony interwał czasu.

WF3. Monitorowanie na telefonie dziecka.
WF3.1. Sprawdzanie położenia.
WF3.1.1. Sprawdzenie terminu lokalizowania.
WF3.1.2. Porównanie z harmonogramem.
WF3.1.2.1. Alarmowanie w przypadku niezgodności.
WF3.2. Alarmowanie w przypadku wykrycia upadku.
WF3.3. Przesyłanie informacji.
WF3.3.1.
WF3.3.2. Sprawdzenie terminu przesłania danych.
WF3.3.3. Przesłanie danych.




Rysunek 11. Monitorowanie na telefonie dziecka

## 5.4 Wymagania niefunkcjonalne

Zbiór tych wymagań opisuje, jakie wymagania wobec systemu mają być spełnione, oprócz wymagań funkcjonalnych. Głównie skupiają się na sposobie pracy, wydajności, bezpieczeństwie itp.

WNF1. Niezawodność, dostępność

System powinien być dostępny w każdy dzień tygodnia (w weekend również, dziecko może mieć zajęcia pozalekcyjne) w godzinach 7-21.

WNF2. Bezpieczeństwo

WNF3. Urządzenia mobilne przesyłają informację tylko na numery telefonów rodziców, a łącząc się z systemem autoryzacji dokonuje się za pomocą loginu i hasła. Połączenie ma być szyfrowane, a także ma być utrzymywana sesja, która będzie stanowić dodatkowy mechanizm zabezpieczający połączenie.Zajętość pamięci oraz zasobów

Rozróżniamy pojemność urządzeń mobilnych, oraz zajętość bazy danych na serwerze. Dane buforowane na urządzeniach mobilnych nie powinny utrudniać pracy (są to dane tekstowe) przez zajęcie zbyt dużej ilości miejsca, dlatego przy braku nawiązania połączenia internetowego w ciągu trzech dni, najstarsze dane będą usuwane w miarę jak pojawiać się będą nowe. Natomiast baza danych na serwerze ma za zadanie przechowywać historię przebiegów dnia przez miesiąc.

WNF4. Wydajność

System powinien w pierwszej wersji obsłużyć 200 użytkowników. Synchronizacja danych z serwerem powinna przebiegać płynnie dla 30 użytkowników podłączonych jednocześnie. Przeglądanie natomiast historii przebiegów dnia bez spadku wydajności powinno być zapewnione dla 50 użytkowników.

WNF5. Zarządzalność

Konfigurowaniem urządzeń i kont zajmuje się rodzić. Po zainstalowaniu i skonfigurowaniu urządzenia dziecka, nie wykonuje ono żadnych akcji (poza połączeniem się z Internetem raz dziennie w celu synchronizacji, sama synchronizacja natomiast odbywa się w tle).

WNF6. Wiarygodność

Zapewnienie szyfrowania połączenia i mechanizmu autoryzacji użytkowników ma zminimalizować liczbę fałszywych alarmów zgłaszanych przez system. Patronus nie jest natomiast w stanie zweryfikować błędów powodowanych błędną pracą m.in. modułu GPS.

WNF7. Użyteczność

System ma spełniać, co najmniej 80% wymagań funkcjonalnych, zwłaszcza dotyczących pracy urządzeń mobilnych.

WNF8. Ergonomia

Użytkowanie systemu powinno być intuicyjne i przejrzyste dla użytkownika. Aplikacja dziecka działa cały czas w tle, dbając o małe zużycie baterii (zwłaszcza dotyczy to korzystania z nadajnika GPS). Aplikacja rodzica również działa w tle. W tryb normalnej pracy włączania jest po otrzymaniu alarmu, lub na życzenie użytkownika.

WNF9. Zrozumiałość

Konstrukcja systemu powinna być wykonana w ten sposób, że po przeczytaniu instrukcji, każdy użytkownik nie miał problemów w jego skonfigurowaniu i późniejszej obsłudze. Główną ideą systemu ma być jego prostota obsługi.

WNF10. Kompletność

Kompletność systemu ma być zapewniona na poziomie głównej funkcjonalności ( w pierwszej wersji systemu), mianowicie monitorowanie położenia dziecka i alarmowanie rodzica w przypadku niezgodności z harmonogramem.

WNF11. Zgodność

Jest to pierwsza wersja systemu – nie rozpatrujemy, zatem zgodności z wcześniejszymi wersjami.

Pierwsza wersja systemu zakłada działanie aplikacji mobilnych na urządzeniach z systemem Windows Phone 7. Natomiast projekt aplikacji mobilnych ma pozwolić na ich prostą implementację w przyszłych wersjach na dowolne platformy urządzeń mobilnych (minimalne wymagania, co do urządzenia to obsługa technologii JAVA). Indywidualizacja

Wygląd aplikacji jest zależny od użytego systemu operacyjnego i przeglądarki internetowej, natomiast sam projekt witryny jest niezależny od użytkownika. Natomiast aplikacja mobilna rodzica zależy od platformy urządzenia ( w pierwszej wersji jest to Windows Phone 7), zatem zmiany wizualne aplikacji ograniczone są do zmiany schematów kolorów systemu operacyjnego.

WNF12. Skalowalność

System przeznaczony jest w pierwszej wersji dla około 200 użytkowników, czyli około 50 rodzin

# 6 Architektura

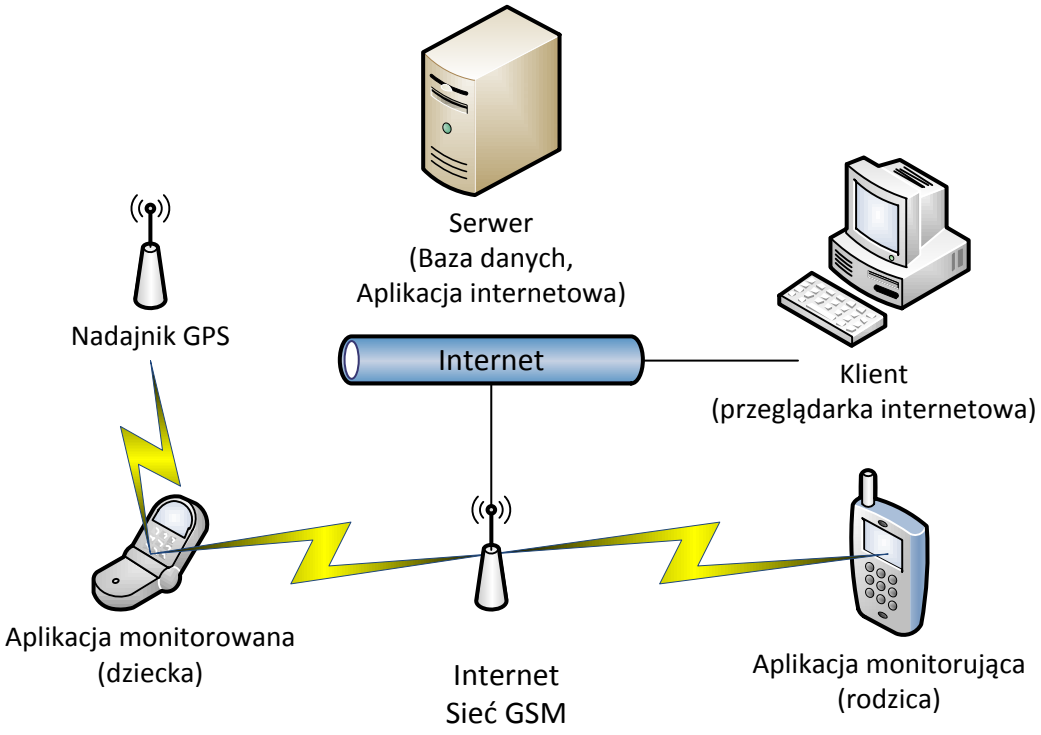
Mamy sprecyzowane już jakie główne funkcje ma spełniać system Patronus zatem teraz pora zastanowić się jak one mają być spełniane. Musimy rozważyć kilka możliwości, zobaczyć, która z metod najlepiej sprawdzi się w naszym przypadku. Także warto rozważyć na tym etapie sposób wykonania całości systemu, gdyż może to narzucić pewne ograniczenia jeśli chodzi o niektóre funkcjonalności.

Domyślną platformą na jakiej będzie działał system jest Windows Phone 7.5 Mango. Aczkolwiek nie ma przeciwskazań do działania systemu na innych systemach mobilnych.

System będzie składał się będzie z urządzeń mobilnych rodziców, dzieci, a także z części serwerowej, która będzie przechowywała informacje konfiguracyjne, oraz będzie odpowiadała za zarządzanie synchronizacją danych. Wniosek z tego jest taki, że potrzebna też będzie w tej części baza danych.

Zatem system Patronus będzie składał się z następujących komponentów:

1. Części centralnej (Patronus Serwer) (serwer + baza danych )
2. Aplikacja internetowej – Patronus Viewer
3. Aplikacji mobilnej rodziców – Patronus Parent.
4. Aplikacji mobilnej dzieci – Patronus Child.




Rysunek 12. Schemat systemu

Projekt systemu zakłada dwie metody przesyłania danych. Podstawowym medium przesyłania informacji jest połączenie internetowe. Dodatkowo natomiast wprowadzone zostanie połączenie SMS'owe dla komunikatów która są najpilniejsze, aby przesyłane były bezpośrednio między aplikacjami mobilnymi, bez udziału serwera, co za tym idzie nie ma potrzeby utrzymywania sesji i odpytywania się serwera o odpowiedź.

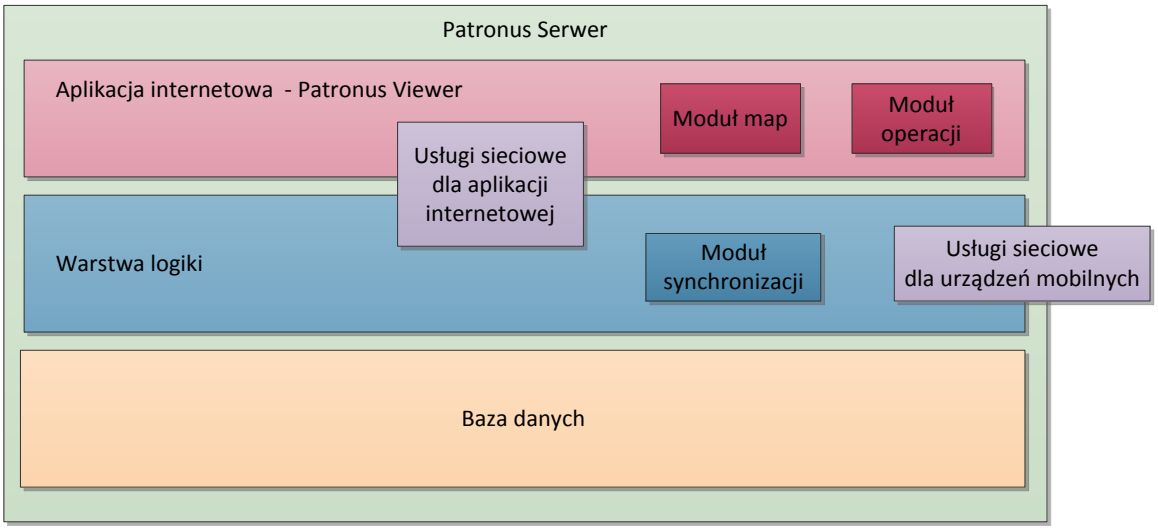
Komunikacja, oparta o SMS'y nie będzie docelowo implementowana w tej wersji systemu, z dwóch przyczyn, po pierwsze jest ona dedykowana na platformę Windows Phone 7.5, która nie udostępnia możliwości wysyłania widomości SMS bez interakcji z użytkownikiem, a po drugie komunikacja SMS'owa jest obecnie wypierana przez połączenia oparte o transmisję danych, więc jest to bardziej przyszłościowe rozwiązanie

Wiadomości SMS będą używane wyłącznie w sytuacjach alarmowych, w których ważną rzeczą jest jak najmniejsze opóźnienie dostarczenia widomości.

## 6.1 Część centralna – Patronus Serwer

Serwer systemu Patronus pełni główne funkcje przechowywania, udostępniania i zarządzania danymi w systemie. Do tych celów przeznaczone są kolejne warstwy serwera:

- Baza danych
- Warstwa logiki
- Aplikacja internetowa – Patronus Viewer




Rysunek 13 Schemat komponentów serwera

### 6.1.1 Baza danych

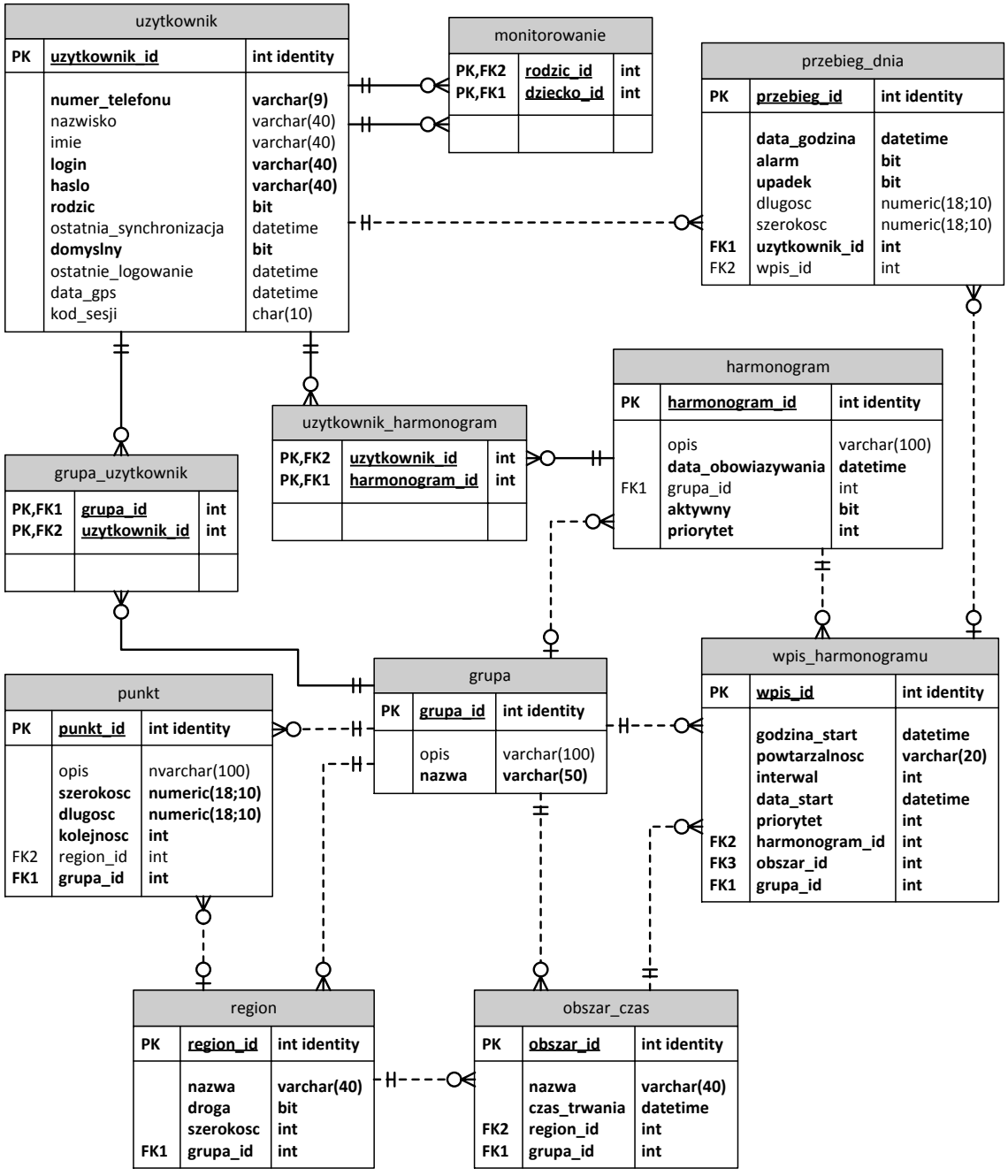
Baza danych jest konieczną częścią systemu, zwłaszcza po stronie serwera, który będzie zbierał i przechowywał wszystkie informacje o jego pracy.

W Patronusie występują grupy użytkowników, które zwykle odpowiadają rodzinie, użytkownicy zorganizowani w grupę mają dostęp do wspólnych danych (utworzony przez jednego użytkownika harmonogram jest widoczny dla drugiego). Natomiast nie jest możliwe używanie danych z innej grupy. Wersja systemu która będzie w mojej pracy tworzona zakłada, że każdy użytkownik może być przypisany wyłącznie do jednej grupy, aczkolwiek baza danych zaprojektowana jest przyszłościowo, żeby w potencjalnych przyszłych wersjach była możliwość by użytkownik mógł działać w ramach różnych grup. W ramach grupy użytkownicy mogą pełnić rolę rodzica bądź też dziecka. Rodzice nadzorują położenie dzieci, natomiast dzieciom przypisywane są harmonogramy zajęć na każdy dzień.

Aby utworzyć harmonogram, najpierw musimy mieć do dyspozycji wyznaczone regiony geograficzne. Każdy wyznaczony region składa się z koła, lub łamanej (drogi). Do takiego

regionu możemy najpierw określić czas jaki dziecko może na nim przebywać. Używając jednego regionu mamy możliwość utworzenia wielu obszarów z przypisanym czasem przebywania. Następnie za pomocą powstałych obszarów możemy już budować harmonogram dla dziecka.

Każdy harmonogram może składać się z wielu obszarów z opisanym dla niego od której godziny obowiązuje i co jaki interwał czasu ma się on powtarzać w harmonogramie (np. co tydzień, lub każdego 5 dnia miesiąca). Tak utworzony harmonogram może być przypisany dla wielu dzieci.




**Rysunek 14 Schemat bazy danych serwera**

Baza ta, ze schematem[1] przedstawionym powyżej, składa się z następujących tabel:

| | **UZYTKOWNIK** |
|---|---|
| **Nazwa pola** | Opis |
| **uzytkownik_id** | Klucz główny tabeli |
| **numer_telefonu** | Numer telefonu komórkowego użytkownika |
| **nazwisko** | Nazwisko użytkownika |
| **imie** | Imię użytkownika |
| **login** | Login użytkownika, używany do jego identyfikacji |
| **haslo** | Hasło użytkownika skojarzone z loginem, w bazie danych przechowywany jest jego skrót (hash) |
| **rodzic** | Bit określający czy użytkownik w systemie pełni funkcję rodzica |
| **ostatnia_synchronizacja** | Data ostatniej synchronizacji danych pomiędzy serwerem, a urządzeniem mobilnym użytkownika |
| **domyślny** | Bit pełni różne funkcje w zależności czy jest to baza danych mobilna (wtedy służy do rozpoznania kto pracuje na urządzeniu), czy też serwerowa (wtedy jest używany do oznaczenia czy dany użytkownik musi pobrać dane z serwera bo uległy zmianie) |
| **ostatnie_logowanie** | Data ostatniego logowania w systemie. W przypadku rodziców jest to data ostatniego pobrania danych o przebiegach dnia dzieci z serwera, a w przypadku dzieci, data ostatniego przesłania ich pozycji |
| **data_gps** | Pole używane jedynie na telefonie dziecka do rozpoznania kiedy był wykonany ostatni odczyt pozycji |
| **kod_sesji** | Kod używany do uwierzytelniania użytkownika i identyfikowania jego sesji. |

| **MONITOROWANIE** | **Tabela tworząca powiązania rodzice-dzieci** |
|---|---|
| **Nazwa pola** | Opis |
| **rodzic_id** | Klucz obcy do rodzica |
| **dziecko_id** | Klucz obcy do dziecka |

| **GRUPA** | **Opisuje grupy użytkowników** |
|---|---|
| **Nazwa pola** | Opis |
| **grupa_id** | Klucz główny |
| **nazwa** | Nazwa grupy |
| **opis** | Opis grupy |

| **GRUPA_UZYTKOWNIK** | **Tabela umożliwiająca relację wiele-do-wielu między grupą a użytkownikami** |
|---|---|
| **Nazwa pola** | Opis |
| **uzytkownik_id** | Klucz obcy do użytkownika |
| **grupa_id** | Klucz obcy do grupy |

| **REGION** | **Podstawowy obiekt opisujący figurę na mapie** |
|---|---|
| **Nazwa pola** | Opis |
| **region_id** | Klucz główny |

[1] Uwagi do schematu: PK – klucz główny, FK – klucz obcy, pogrubienie nazwy kolumny = kolumna wymagana, połączenia między tabelami przy zastosowaniu tzw. kurzych łapek, oznaczają relację jeden do wielu, przy czym okrąg przy stronie tabeli symbolizuje opcjonalność tej strony w relacji

| | |
|---|---|
| **nazwa** | Nazwa regionu |
| **droga** | Bit określający czy dany region opisuje drogę (łamaną), czy też miejsce (koło) |
| **szerokość** | W przypadku drogi to faktycznie jest jej szerokość, natomiast w przypadku miejsca, jest to średnica |
| **grupa_id** | Klucz obcy do grupy |

| **PUNKT** | **Obiekt podrzędny do regionu, opisuje pojedynczy punkt geograficzny, w przypadku miejsc, do regionu przypisany jest tylko jeden punkt** |
|---|---|
| **Nazwa pola** | Opis |
| **punkt_id** | Klucz główny |
| **opis** | Opis punktu |
| **szerokość** | Szerokość geograficzna |
| **dlugosc** | Długość geograficzna |
| **kolejność** | W przypadku budowania drogi (łamanej) jest to numer kolejnego tworzącego ją punktu |
| **region_id** | Klucz obcy do regionu |
| **grupa_id** | Klucz obcy do grupy |

| **OBSZAR_CZAS** | **Tabela powstająca z regionu, oraz określająca ile czasu w danym regionie dziecko będzie spędzać. Możliwe jest wielokrotne tworzenie obszarów z regionu.** |
|---|---|
| **Nazwa pola** | Opis |
| **obszar_id** | Klucz główny |
| **nazwa** | Nazwa obszaru |
| **czas_trwania** | Czas jaki będzie spędzany w tym obszarze |
| **region_id** | Klucz obcy do regionu |
| **grupa_id** | Klucz obcy do grupy |

| **HARMONOGRAM** | **Główny obiekt służący do grupowania planów dnia dzieci** |
|---|---|
| **Nazwa pola** | Opis |
| **harmonogram_id** | Klucz główny |
| **opis** | Opis harmonogramu |
| **data_obowiazywania** | Termin od jakiego harmonogram staje się aktywny |
| **aktywny** | Bit określający czy harmonogram już jest aktualny |
| **grupa_id** | Klucz obcy do grupy |
| **priorytet** | Pole służące do rozwiązywania konfliktów przy wybieraniu odpowiedniego harmonogramu |

| **WPIS_HARMONOGRAMU** | **Obiekty grupowane w harmonogramie, czyli jego pojedyncze wpisy określające zajęcia w ciągu dnia** |
|---|---|
| **Nazwa pola** | Opis |
| **wpis_id** | Klucz główny |
| **godzina_start** | Godzina rozpoczęcia zajęć |
| **powtarzalność** | przyjmuje wartości: „Dzień”, „Tydzień”, „Miesiąc”, „Brak”, mówiące co jaki interwał zajęcia będą się powtarzać |
| **interwal** | Liczba mówiąca co ile dni/tygodni itp. zajęcia się powtarzają |
| **data_start** | Data rozpoczęcia zajęć |
| **priorytet** | Pole służące do rozwiązywania konfliktów przy wybieraniu |

| | |
|---|---|
| | odpowiedniego harmonogramu |
| **harmonogram_id** | Klucz obcy do harmonogramu |
| **obszar_id** | Klucz obcy do obszaru |
| **grupa_id** | Klucz obcy do grupy |

| **UZYTKOWNIK_HARMONOGRAM** | **Tabela pozwalająca na relację wiele-do-wielu pomiędzy użytkownikami i harmonogramami** |
|---|---|
| **Nazwa pola** | Opis |
| **uzytkownik_id** | Klucz obcy do użytkownika |
| **harmonogram_id** | Klucz obcy do harmonogramu |

| **PRZEBIEG_DNIA** | **Tabela przechowująca informacje o przebiegu dnia dzieci, a dokładnie o odczytach pozycji** |
|---|---|
| **Nazwa pola** | Opis |
| **przebieg_id** | Klucz główny |
| **data_godzina** | Data i godzina odczytu pozycji |
| **alarm** | Czy odczyt był wykonany poza zaplanowanym obszarem |
| **upadek** | Czy odczyt był wywołany zbyt szybkim upadkiem telefonu |
| **dlugosc** | Długość geograficzna |
| **szerokość** | Szerokość geograficzna |
| **wpis_id** | Klucz obcy do wpisu harmonogramu |
| **uzytkownik_id** | Klucz obcy do użytkownika |
| **grupa_id** | Klucz obcy do grupy |

### 6.1.2 Warstwa logiki

Warstwa logiki serwera odpowiada za komunikację z główną bazą danych systemu, sprawdzanie poprawności danych, udostępnianie usług sieciowych do komunikacji z aplikacją internetową oraz urządzeniami mobilnymi.

Składa się z poniższych modułów:

- Moduł synchronizacji danych
- Moduł usług sieciowych
  - Usługi aplikacji internetowej – Patronus Viewer
  - Usługi aplikacji mobilnej rodzica – Patronus Parent
  - Usługi aplikacji mobilnej dziecka – Patronus Child

#### *6.1.2.1 Moduł synchronizacji danych*

Jedną z dwóch głównych funkcjonalności systemu jest synchronizowanie danych telefonów komórkowych z serwerem. Sama synchronizacja jest uruchamiana przez użytkownika, do systemu Patronus należy jedynie przypominanie o niej. Dane które są synchronizowane pomiędzy urządzeniami mobilnymi a serwerem to dane które są niezbędne do pracy urządzeniom mobilnym. Są to:

- Dane użytkownika (użytkowników) – dane użytkownika obsługującego aplikację, plus dane dzieci w przypadku aplikacj Patronus Parent
- Dane punktów, regionów, obszarów, harmonogramów razem z wpisami.

Urządzenia mobilne tak jak i serwer będą miały informacje na temat ostatniego terminu synchronizacji (wyjątkiem są dopiero co zainstalowane aplikacje mobilne). Działające w tle

usługi na telefonach sprawdzać będą czy minęła już doba od ostatniej synchronizacji i wyświetlą wtedy informację o tym, że należy wykonać synchronizację danych. Samą zaś synchronizację musi zapoczątkować użytkownik (przykładowo chce wykonać ją w domu gdzie ma dostęp do sieci WiFi).

Przebieg synchronizacji:

1. Urządzenie mobilne informuje serwer o chęci rozpoczęcia synchronizacji z nim danych.
2. Wysyłane jest zapytanie czy nastąpiły zmiany w danych dotyczących użytkownika telefonu, jeśli tak to wykonywany jest krok 3, a jeśli nie to krok 7.
3. Kolejno synchronizowane są informacje o użytkownikach, regionach, punktach, obszarach, harmonogramach, oraz wpisach harmonogramów.
4. Serwer przy każdej synchronizacji wybiera ze swojej bazy danych informacje jakie są konieczne do przesłania danemu użytkownikowi.
5. Po otrzymaniu kolejnych danych, urządzenie mobilne usuwa swoje stare dane i zastępuje je nowymi.
6. Aplikacja mobilna nie prosi serwera o przesłanie kolejnych danych przed otrzymaniem poprzednich, ma to zminimalizować czas oczekiwania na dane z serwera i zapewnić odpowiednią kolejność synchronizowania danych.
7. Po zakończeniu procedury serwer i aplikacja mobilna oznaczają datę rozpoczęcia procesu synchronizacji, jako datę synchronizacji danych dla użytkownika używającego tego telefonu.

#### *6.1.2.2 Moduł usług sieciowych*

Mówiąc o przepływie danych bardzo ważną rzeczą jest wyspecyfikowania metod za pomocą których elementy systemu będą się ze sobą komunikować. Mowa tutaj o usługach sieciowych, które stanowią główny kanał komunikacyjny.

Możemy wyróżnić dwie usługi sieciowe:

- Patronus Viewer Service - wykorzystywaną przez aplikację zarządzającą systemem – Patronus Viewer
- Patronus Parent Service – używaną przez aplikację rodzica
- Patronus Child Service – używaną przez aplikację dziecka

Podział ten wprowadzi częściowo redundantne metody sieciowe, jednak pozwoli na przejrzyste zarządzanie systemem, oraz łatwiejszą prawdopodobną jego przyszłą rozbudowę. Ponadto część metod w przypadku urządzeń mobilnych powinna zwracać dane w mniejszej ilości, niezbędnej do pracy aplikacji mobilnych, bądź też przekazywać je w innym formacie.

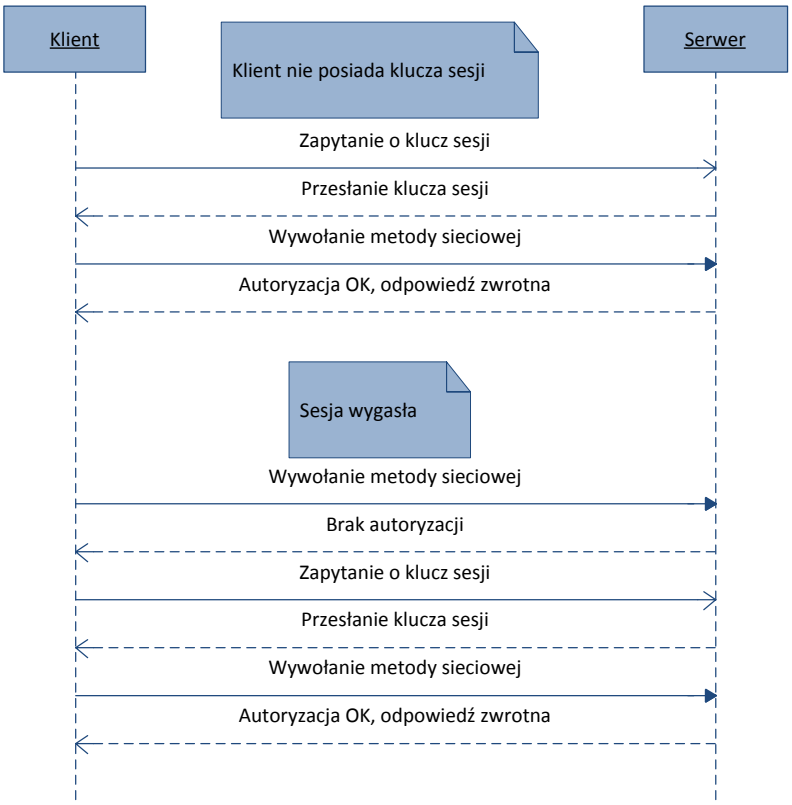
##### 6.1.2.2.1 Uwierzytelnianie i Autoryzacja

Bardzo ważnym aspektem działania systemu jest zapewnienie uwierzytelnienia użytkowników, aby nie możliwe było uzyskanie danych z serwera z poza systemu, oraz autoryzacji by zdecydować czy użytkownik jest uprawiony do dostępu do metody sieciowej. Udostępnianie usług sieciowych powinno odbywać się poprzez szyfrowane połączenie SSL, by ograniczyć możliwość podsłuchania przesyłanych danych.

Użycie uwierzytelniania użytkowników przez serwer, wymagałoby utworzenia grup i ról użytkowników na serwerze (przykładowo ASP.NET) co powodowałoby dublowanie danych przechowywanych w bazie danych.

Zatem rozwiązanie tutaj zastosowane będzie nieco inne. Mianowicie będzie utrzymywana sesja połączenia. Każda sesja będzie posiadała swój identyfikator sesji – losową liczbę generowaną przez serwer podczas nawiązywania połączenia przez urządzenie mobilne. Identyfikator ten również może być nazywany kluczem sesji. Klucz ten będzie powiązany z konkretnym użytkownikiem. Po otrzymaniu klucza sesji, użytkownik odsyła na serwer skrót (hash) wyliczony z połączonego wspomnianego klucza sesji i skrótu hasła użytkownika (przechowywanego lokalnie na urządzeniu mobilnym. Analogiczną operację wykonuje serwer i porównuje wynik z danymi otrzymanymi od użytkownika. Jeśli skróty te są takie same, uwierzytelnienie użytkownika przebiegło pomyślnie i mamy nawiązaną sesję połączenia.

Następnie do każdego wywołania metody usługi sieciowej będą dołączana dwa parametry: identyfikator użytkownika i skrót z klucza sesji. Będzie przechowywana również informacja o godzinie ostatniego logowania się użytkownika na serwerze. Odpowiedni timer będzie dbał o to, że gdy użytkownik nie logował się zbyt długo (5 minut) to klucz sesji wygaśnie.




**Rysunek 15 Schemat wywołań metod z usług sieciowych serwera**

Ogólny algorytm uwierzytelniania użytkownika przy wywołaniu usługi sieciowej przedstawia się następująco (dla skrócenia zapisu aplikacja mobilna będzie nazywana klientem):

- Klient sprawdza czy posiada lokalnie zapisany kucz sesji, jeśli tak to wywołuje metodę sieciową dołączając do niej swój identyfikator oraz wyliczony skrót z połączonego skrótu hasła z kluczem sesji.
- Serwer zawsze najpierw dokonuje porównania wyliczonego skrótu z otrzymanym. Klient i serwer używają tej samej funkcji skrótu. W przypadku gdy dane wyliczone przez serwer

nie zgodzą się z otrzymanymi, lub też serwer stwierdzi że klucz sesji wygasł odsyłany jest komunikat o braku autoryzacji.

- Klient w przypadku gdy autoryzacja się nie udała prosi serwer o nowy klucz sesji, po czym rozpoczyna procedurę od początku.

Poniżej znajduje się opis metod udostępnianych przez usługi sieciowe. Dla przejrzystości zapisu nie są do nich dodawane parametry służące do uwierzytelnienie (czyli identyfikator użytkownika i klucz sesji).

#### 6.1.2.2.2 Patronus Viewer Service

Ta usługa sieciowa jest przeznaczona do komunikacji z aplikacją Patronus Viewer i udostępnia następujące metody:

| Nazwa metody | Parametry | Typ zwracany | Opis |
|---|---|---|---|
| GetSessionKey | identyfikator użytkownika | Klucz sesji | Metoda służąca do nawiązania sesji połączenia z serwerem. Serwer tworzy losową liczbę która zostaje kluczem sesji. |
| Login | identyfikator użytkownika, skrót hasła i klucza sesji | uzytkownik – obiekt reprezentujący użytkownika, lub NULL | Służy do zalogowania się w systemie, jeśli proces uwierzytelnienia przebiegnie pomyślnie, zostanie zwrócony obiekt reprezentujący zalogowanego użytkownika, a w przeciwnym wypadku NULL. |
| GetUsersByGroupId | groupId – identyfikator grupy | Lista użytkowników | Zwraca listę użytkowników dla zadanej grupy (rozpoznawanej po jej identyfikatorze przekazanym w parametrze). Do każdego użytkownika dołączone są dodatkowo informacje o tym jakie dzieci on monitoruje (w przypadku rodziców), lub przez jakich rodziców jest monitorowany (w przypadku dzieci). |
| AddUserAndGroup | entity – dane użytkownika, group – dan grupy | Komunikat o błędzie, lub NULL | Dodaje do bazy użytkownika i grupę. Służy do rejestracji nowego użytkownika, która powoduje automatycznie utworzenie nowej grupy użytkowników. W przypadku gdy próbujemy dodać użytkownika o istniejącej w bazie nazwie użytkownika (ang.login) lub telefonie powinien być zwrócony komunikat o błędzie, natomiast gdy wszystko przebiegnie pomyślnie, zwracamy NULL. |
| AddUser | entity – dane użytkownika, groupId – identyfikator grupy | Komunikat o błędzie, lub NULL | Dodaje do bazy użytkownika, do określonej w parametrze grupy (rozpoznawanej po identyfikatorze). Metoda ta powinna być używana w przypadku gdy zalogowany użytkownik dodaje konto kolejnego użytkownika, tym samym przypisując go do grupy w której sam jest. |

| | | | |
|---|---|---|---|
| SaveUser | entity – dane użytkownika | Komunikat o błędzie, lub NULL | Zapisuje w bazie zmiany w informacjach na temat użytkownika. Podobnie jak w przypadku dodawania nowego użytkownika powinno być wykonanie sprawdzenia czy w bazie nie istnieje już użytkownik z takim loginem lub telefonem. Nie można zmienić użytkownikowi nazwy (loginu) lub telefonu na taki jak ma inny użytkownik. |
| GetUser | tel – numer telefonu użytkownika (unikalny w systemie) | uzytkownik | Zwraca użytkownika o zadanym numerze telefonu. |
| DeleteUser | entity – dane użytkownika | brak | Usuwa z systemu profil przekazanego jako parametr użytkownika. Należy upewnić się przed wywołaniem tej metody, czy nie próbujemy usunąć użytkownika na którego konto jesteśmy teraz zalogowani. |
| AddPlan | plan – dane o harmonogramie,<br>groupId – identyfikator grupy, | harmonogram | Dodawanie nowego harmonogramu. Każdy świeżo dodany harmonogram powoduje równocześnie zapisanie powiązanych z nim wpisów harmonogramu. Harmonogram po dodaniu jest też dopisywany do grupy z identyfikatorem podanym w parametrze, oraz przypisywany do konkretnego użytkownika, którego identyfikator również przekazywany jest w parametrze. |
| GetAllPlansForGroup | groupId – identyfikator grupy | Lista harmonogramów | Zwraca listę harmonogramów dotyczących konkretnej grupy użytkowników. Po stronie aplikacji zarządzającej lista wszystkich harmonogramów w grupie potrzebna będzie przy wykorzystywaniu istniejących harmonogramów do tworzenia planów dnia użytkowników, którzy mogą z nich skorzystać. |
| GetAllPlansForUser | | Lista harmonogramów | Zwraca listę harmonogramów do jakich jest przypisany użytkownik o zdanym w parametrze wywołania identyfikatorze. |
| AddPlanToUser | plan – dane o harmonogramie,<br>user – dane o użytkowniku | brak | Pozwala na dopisanie istniejącego harmonogramu do użytkownika. Harmonogram i użytkownik powinny być przekazane jako parametry metody. Nie powinno być możliwości dwukrotnego dodania tego samego harmonogramu do użytkownika |

| | | | |
|---|---|---|---|
| DeletePlanFromUser | plan – dane o harmonogramie,<br>user – dane o użytkowniku | brak | Metoda dokonująca operacji przeciwnej do metody powyższej. Pozwala na usunięcie powiązanego harmonogramu z użytkownikiem. Gdy usuniemy harmonogram powiązany z użytkownikiem i nie będzie on powiązany z żadnym innym użytkownikiem to będzie on na stałe usuwany z systemu. |
| **GetPlacesByGroup** | groupId – identyfikator grupy | Lista obszarów | Zwraca listę obszarów, wraz z powiązanymi z nimi regionami. Wszystko w ramach jednej grupy użytkowników |
| GetRegionsByGroup | groupId – identyfikator grupy | Lista regionów | Zwraca listę istniejących w ramach grupy regionów. Każdy region powinien być powiązany z listą punktów, które go dokładnie definiują. Lista ta powinna być dołączona do każdego zwracanego regionu. |
| AddRegion | entity – dane o regionie | region | Dodaje do systemu zdefiniowany przez użytkownika region. Powoduje to również dodanie wszystkich powiązanych z nim punktów. Każdy z dodawanych obiektów jest przypisywany do konkretnej grupy użytkowników. Między grupą a regionami i punktami istnieje relacja jeden-do-wielu, dzięki czemu identyfikator grupy powinien być zawarty w przekazywanym jako parametr regionie (klucz obcy grupa_id). |
| AddPlaceByRegion | entity – dane o obszarze,<br>regId – identyfikator regionu | obszar_czas | Dodanie obszaru definiującego czas przebywania osoby na zadanym regionie. Region rozpoznawany jest poprzez identyfikator przekazany jako parametr metody. |
| DeleteRegion | entity – dane o regionie | true/false | W przypadku gdy dany region nie jest używany do tworzenia obszaru jest on usuwany, wtedy metoda zwraca TRUE, a przeciwnym wypadku nie usuwa regionu i zwraca FALSE. |
| DeletePlace | entity – dane o obszarze | true/false | Kiedy dany obszar nie jest używany w żadnym wpisie harmonogramu jest on usuwany I metoda zwraca TRUE. W przeciwnym wypadku zwracane jest FALSE i obszar pozostaje w bazie danych. |
| GetDayEvents | date – data | Lista przebiegów dnia | Zwraca listę zdarzeń (wpisów z przebiegu dnia) dziecka o zadanym ID i dla dnia określonego w parametrze *date*. |

| | | | |
|---|---|---|---|
| SetUserNeedSynch | context – kontekst dostępu do bazy danych, groupId – identyfikator grupy | brak | Metoda pomocnicza, nie wystawiana na zewnątrz z serwisu. Ma za zadanie podczas zmian dokonywanych na harmonogramie i użytkownikach w ramach grupy, oznaczanie, że po stronie serwera dane dla użytkowników tej grupy się zmieniły (wykorzystywane przez usługi dla urządzeń mobilnych). |

#### 6.1.2.2.3 Patronus Parent Service

Usługa sieciowa służy do komunikacji pomiędzy serwerem a aplikacją Patronus Parent, oraz udostępnia w tym celu poniższe metody:

| Nazwa metody | Parametry | Typ zwracany | Opis |
|---|---|---|---|
| GetSessionKey | identyfikator użytkownika | Klucz sesji | Metoda służąca do nawiązania sesji połączenia z serwerem. Serwer tworzy losową liczbę która zostaje kluczem sesji. |
| Login | identyfikator użytkownika, skrót hasła i klucza sesji | uzytkownik – obiekt reprezentujący użytkownika lub NULL | Metoda dla urządzeń mobilnych odpowiadająca za logowanie się do systemu. Serwer sprawdza poprawność z wyliczonym przez siebie skrótem z hasła użytkownika i aktualnego klucza sesji. |
| IsSynchNeeded | | true/false | Służy do sprawdzenia czy dany użytkownik potrzebuje pobrać dane z serwera. Dane na serwerze po każdej zmianie są sygnalizowane odpowiednią flagą w rekordach wszystkich użytkowników w ramach grupy. Flaga zmieniana jest na informującą o braku konieczności synchronizacji danych, po przesłaniu ich użytkownikowi na telefon. |
| GetChildren | | Lista użytkowników | Dla zadanego klucza głównego wyszukiwany jest użytkownik, a następnie wszystkie dzieci których monitorowaniem jego aplikacja się zajmuje. |
| GetRegions | | Lista regionów | Dla zadanego użytkownika pobiera wszystie regiony jakie są dostępne w ramach jego domyślnej grupy. |
| GetPoints | | Lista punktów | Dla zadanego użytkownika pobiera wszystkie punkty jakie są dostępne w ramach jego domyślnej grupy. |
| GetPlaces | | Lista obszarów | Dla zadanego użytkownika pobiera wszystkie obszary jakie są dostępne w ramach jego domyślnej grupy. |
| GetPlans | | Lista harmonogramów | Dla zadanego użytkownika pobiera wszystkie harmonogramy jakie są dostępne w ramach jego domyślnej |

| Nazwa metody | Parametry | Typ zwracany | Opis |
|---|---|---|---|
| | | | grupy. |
| GetPlanEntries | | Lista wpisów harmonogramu | Dla zadanego użytkownika pobiera wszystkie wpisy harmonogramu jakie są dostępne w ramach jego domyślnej grupy. |
| GetDayEvents | dateFrom - data lub NULL | Lista przebiegów dnia | Dla zadanego użytkownika zwraca przebiegi dnia dzieci których monitorowaniem jego aplikacja się zajmuje. Zwraca dane od terminu określonego jako parametr metody. Gdy parametr ten nie zawiera informacji (NULL), zwraca listę wszystkich zdarzeń z aktualnego dnia. |
| GetUsersAndPlans | | Słownik który zawiera jako klucz identyfikator użytkownika, a jako wartość listę identyfikatorów harmonogramów | Dla zadanego użytkownika pobiera wszystkie zależności pomiędzy użytkownikami i harmonogramami. Zwraca słownik, który jako klucze zawiera identyfikatory dzieci, a jako wartości, listy identyfikatorów harmonogramów do jakich dzieci są przypisane. |
| SetSynchDone | synchDate – data rozpoczęcia synchroniza-cji | brak | Metoda wywoływana na zakończenie synchronizacji. Zapisuje w parametrach użytkownika datę ostatniej synchronizacji danych, oraz zmienia flagę informującą o konieczności przesłania danych z serwera na telefon na nieaktywną. |

#### 6.1.2.2.4 Patronus Child Service

Usługa sieciowa pozwala na komunikację pomiędzy serwerem a aplikacją dziecka Patronus Child.

Do tej komunikacji używane są następujące metody:

| **Nazwa metody** | **Parametry** | **Typ zwracany** | **Opis** |
|---|---|---|---|
| GetSessionKey | identyfikator użytkownika | Klucz sesji | Metoda służąca do nawiązania sesji połączenia z serwerem. Serwer tworzy losową liczbę która zostaje kluczem sesji. |
| LoginChild | tylko domyślne | uzytkownik – obiekt reprezentujący użytkownika, lub NULL | Służy do zalogowania się na profil dziecka podczas konfiguracji urządzenia mobilnego. |
| GetChildRegions | | Lista regionów | Pobiera regiony, które są używane w harmonogramie zadanego dziecka. Nie ma potrzeby |

| | | | |
|---|---|---|---|
| | | | przesyłania danych na telefon dziecka z których on nie korzysta. |
| GetChildPoints | | Lista punktów | Zwraca listę punktów wykorzystywanych przez regiony używane w harmonogramie dziecka. |
| GetChildPlaces | | Lista obszarów | Analogicznie jak poprzednie metody, zwraca obszary dla zadanego dziecka. |
| GetChildPlans | | Lista harmonogramów | Analogicznie jak poprzednie metody, zwraca harmonogramy dla zadanego dziecka. |
| GetChildPlanEntries | | Lista wpisów harmonogramu | Analogicznie jak poprzednie metody, zwraca wpisy harmonogramu dla zadanego dziecka. |
| SendDayEvents | entities – lista przebiegów dnia,<br>time – godzina wysłania | godzina przesłania danych | Metoda pozwalająca aplikacji dziecka przesłanie informacji o przebiegu dnia. Dane przesyłane są w postaci listy wpisów tabeli *przebieg_dnia*. Oprócz tego przesyłana jest godzina, żeby było wiadomo do jakiego czasu wpisy te obowiązują (czasem przesłanie danych może odbywać się z opóźnieniem). |

### 6.1.3 Warstwa prezentacji – Patronus Viewer

Za prezentację danych użytkownikowi odpowiadać będzie aplikacja internetowa, która też pozwalać będzie na konfigurację systemu, zarządzanie profilami użytkowników, oraz harmonogramów dnia dzieci, oraz przeglądanie historii zmian położenia dzieci.

Udostępniać ona będzie następujące strony:

1. Ekran powitalny
2. Ekran logowania i rejestracji – pozwalający zalogować się w systemie na już utworzony profil użytkownika (tylko rodzice mogą się tutaj logować), oraz ekran dodawania nowego użytkownika, który automatycznie tworzy też nową grupę użytkowników. Aby użytkownicy byli w tej samej grupie należy tworzyć profile kolejnych użytkowników po zalogowaniu na już utworzony profil.
3. Zarządzanie – ekran dodawania nowych profili użytkowników w systemie, tworzenia relacji, który rodzic ma monitorować położenie którego dziecka, oraz zarządzanie planami dnia dzieci, przypisywanie ich do konkretnych profili dzieci, a także przeglądanie planu dnia na wybrany dzień.

4. Harmonogramy – ekran który umożliwia tworzenie obszarów i regionów w widoku mapy i ich przeglądanie.
5. Historia – ekran który dla zadanego profilu dziecka i daty wyświetla informacje o zmianach położenia, oraz umożliwia ich wyświetlenie na mapie.

#### 6.1.3.1 Moduł operacji

Moduł ten odpowiada za komunikację za pomocą usług sieciowych z serwerem, a także na zarządzanie danymi całej aplikacji, przechowywanie pobranych z serwera danych w pamięci operacyjnej, oraz nadzorowanie wyświetlania ekranów. Realizowany będzie według wzorca Singleton.

#### 6.1.3.2 Algorytmy

Algorytmami jakie będzie wykorzystywać aplikacja internetowa są algorytmy wyznaczania obszarów na mapie według zadanych punktów.

Przy wyświetlaniu definiowanych przez użytkownika obszarów na mapie konieczne jest wyznaczenie okręgu dookoła niego (w przypadku miejsc), lub wyznaczenie punktów składających się na obszar wokół drogi o zadanej szerokości.

##### 6.1.3.2.1 Miejsce (koło)

Zaczynając od miejsca, wiemy, że składać się ono ma z punktu środka okręgu, oraz promienia. Środek okręgu określany jest we współrzędnych geograficznych, a promienia w metrach. Moduły obsługujące mapy nie udostępniają możliwości wykreślenia okręgu na mapie w ten sposób, by rozmiar okręgu był przypisany do konkretnych współrzędnych geograficznych. Wyświetlany rozmiar okręgu nie jest skalowany przy powiększaniu/pomniejszaniu mapy. Konieczne jest wyznaczenie współrzędnych punków na okręgu, co jest oczywiście obarczone pewnym przybliżeniem. Jednak zakładając, że okrąg wokół punktu będzie się składał z 360 punktów, oraz zakładając, że promień okręgu nie będzie przekraczał ok. 1000 metrów, to takie przybliżenie w zupełności wystarczy i oko ludzkie nie będzie w stanie określić czy ma do czynienia z faktycznym okręgiem wykreślonym na mapie, czy też z 360-kątem.

Możliwe jest jednak przy dużym promieniu i wysokiej skali mapy rozpoznanie linii prostych, pomimo tego faktu uważam, że zastosowanie przybliżonych punktów które będą się składać na okrąg jest dobrym rozwiązaniem.

Współrzędne geograficzne możemy założyć, że będą podane w stopniach lub radianach, jednak promień będzie w metrach, natomiast wynik jaki chcemy otrzymać to lista 360 punktów również jako współrzędne geograficzne. Zgłębiając problem natrafiłem na pewien wpis na blogu [9], którego autor Chris Pietschmann napotkał taki sam problem. Pozwolę sobie przytoczyć zastosowane przez niego rozwiązanie.

**Stałe:**

***earthRadius*** – promień Ziemi w kilometrach – 6376 km.

**Zmienne:**

***lat,lng*** – zmienne określające odpowiednio szerokość (latitude) i długość (longitude) geograficzną, wyrażoną w radianach

***x*** – zmienna przyjmująca wartości od 0 do 360 (decydująca który teraz punkt na okręgu będziemy wyznaczać, 0 oznacza punkt na północ od środka okręgu)

***brng*** – zmienna powstała z przekształcenia zmiennej x na radiany, wg wzoru

$$brng = {}^{x\pi}/_{180}$$

***r*** – promień okręgu

***d*** – odległość kątowa pokrywająca obszar od środka okręgu do jego obwodu na powierzchni Ziemi, powstaje z podzielenia promienia okręgu (w kilometrach) przez promień Ziemi.

**Wynik:**

***latRadians, lngRadians*** – odpowiednio szerokość i długość geograficzna w radianach

**Obliczenia:**

$$\boldsymbol{latRadians = \sin^{-1}(\sin lat \cdot \cos d + \cos lat \cdot \sin d \cdot \cos brng)}$$

$$\boldsymbol{lngRadians = lng + \tan^{-1}\left(\frac{\sin brng \cdot \sin d \cdot \cos lat}{\cos d - \sin lat \cdot \sin latRadians}\right)}$$

Zastosowanie tego algorytmu pozwala wygodnie wyznaczyć punkty poszukiwanego przez nas okręgu.

#### 6.1.3.2.2 Droga

Drugi z możliwych sposobów opisuje drogę pomiędzy miejscami w postaci listy kolejnych współrzędnych geograficznych tworzących łamaną, oraz parametru który mówi jak szeroko wokół kolejnych odcinków roztacza się obszar.

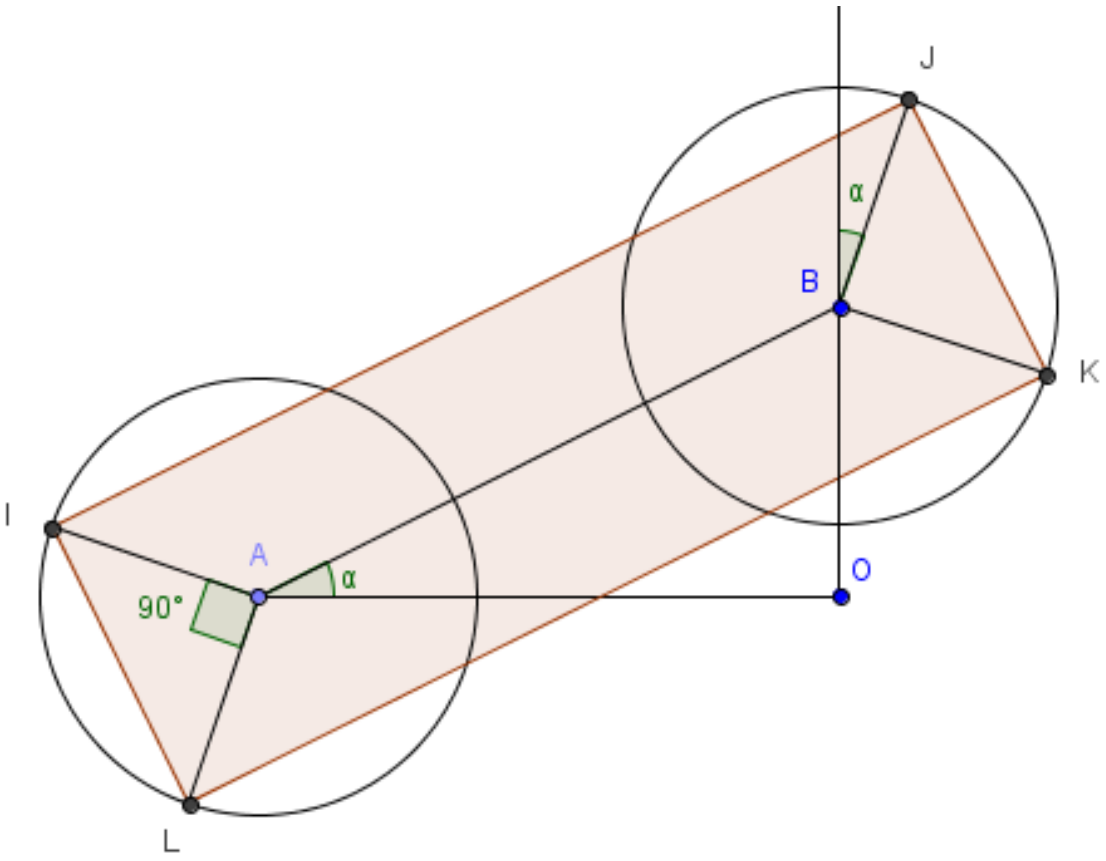
Możemy rozważyć 3 szczególne przypadki takiej drogi:

1. Składająca się z jednego punktu – a co za tym idzie obszarem wokół punktu będzie kwadrat o długości boku zadanej w parametrze i równoległych do głównych kierunków geograficznych.
2. Składająca się z dwóch punktów – co powoduje powstanie wokół nich obszaru w postaci prostokąta. Sytuację tą ilustruje schemat z rysunku 16.

*Uwagi do schematu:*

- Punktami wyjściowymi są punkty A i B, składowane w bazie danych, które jako współrzędne przyjmują długość i szerokość geograficzną
- Obszar wokół drogi wyznaczany jest przez wierzchołki IJKL, które tworzą prostokąt
- Parametry wymagane do zastosowania algorytmu wyliczania współrzędnych punktów na okręgu, opisanego w poprzednim podrozdziale (posłuży on do wyznaczenia wierzchołków prostokąta, dwa leżą na okręgu wokół pierwszego punktu (A), a dwa wokół drugiego(B)):
    - Promień okręgu wokół zadanego punktu, r = $\frac{a\sqrt{2}}{2}$, gdzie a – szerokość drogi

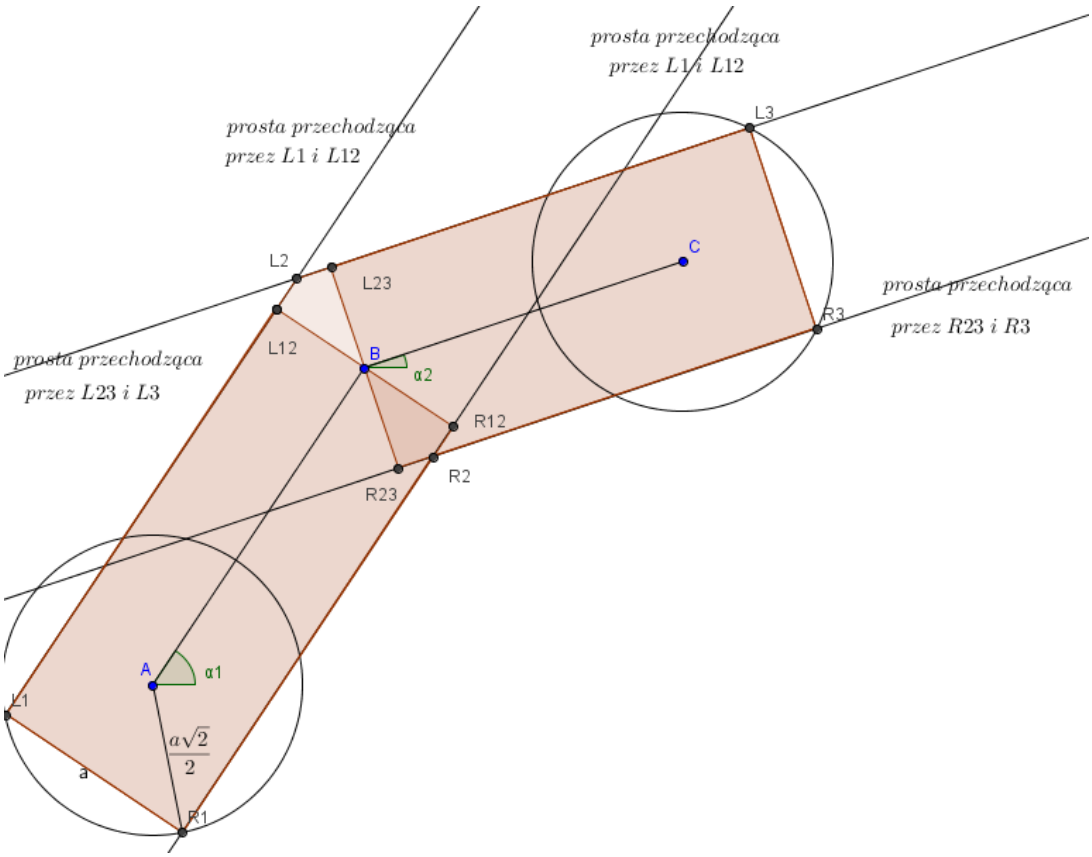
- Współrzędna kątowa (x), określająca który punkt na okręgu ma być wyliczany (0 stopni oznacza kierunek północny, i rośnie zgodnie z kierunkiem wskazówek zegara). Pomocniczo wprowadzam:
  - kąt $\alpha$ (OAB) - nachylenie do poziomu pomiędzy punktami A i B, czyli kąt o jaki prostokąt jest obrócony względem poziomu w kierunku przeciwnym do wskazówek zegara
- Wartości kąta x w przypadku gdy punkty A i B leżą na jednej szerokości geograficznej ($\alpha = 0$):
  - Dla punktu I : $x = -45°$
  - Dla punktu L: $x = 225°$
  - Dla punktu J: $x = 45°$
  - Dla punktu K: $x = 135°$
- Wartość kąta x w zależności od wyliczanego wierzchołka:
  - Dla punktu I: $x = \alpha - 45°$
  - Dla punktu L: $x = \alpha + 225°$
  - Dla punktu J: $x = \alpha + 45°$
  - Dla punktu K: $x = \alpha + 135°$




Rysunek 16 Obszary wokół dwóch punktów drogi

3. Składająca się z trzech lub więcej punktów – wtedy wokół prostej powstaje wielokąt, który tworzony jest z kolejnych punktów leżących po obu stronach drogi.

Pierwsze dwa i ostatnie dwa punkty (na rysunku L1, R1 i L3, R3), tworzone są według algorytmu z punktu 2.




**Rysunek 17 Obszar wokół drogi składającej się z trzech punktów ABC**

Wyznaczenie natomiast pozostałych punktów opisuje następujący algorytm:

- Rozpatrujemy kolejne odcinki łamanej
- W każdej iteracji wyliczamy współrzędne leżące na prostych prostopadłych do odcinka i przechodzących odpowiednio przez pierwszy i drugi punkt odcinka. Wyliczone 4 punkty (2 przy jednym i dwa przy drugim końcu odcinka) są oddalone od niego o parametr określający szerokość drogi podzielony przez 2 (szerokość drogi to odległość od jednego z wyznaczonych punktów do drugiego).
- Wyjątkiem są wspomniane wcześniej dwa punkty startowe i dwa końcowe, przy wyliczaniu których stosowany jest algorytm z punktu 2.
- Każda kolejna iteracja poza pierwszą korzysta ze współrzędnych punktów wyznaczonych w poprzedniej iteracji
- Mając dane 4 współrzędne z poprzedniej iteracji i 4 z obecnej wyznaczamy równania dwóch prostych przechodzących przez punkty z lewej strony poprzedniej iteracji i obecnej iteracji, oraz kolejne dwie proste dla prawej strony odcinka.
- Znajdujemy punkt przecięcia prostych z lewej strony, oraz prostych z prawej strony.
- Znalezione dwa punkty stają się kolejnymi punktami budowanego wielokąta wokół drogi.
- UWAGA: gdy kolejny odcinek prostej jest skierowany na zachód (w stronę malejących długości geograficznych) konieczne jest zamienienie punktów z lewej strony prostej i prawej, gdyż algorytm wyznaczania punktów wokół zadanego punktu przyjmuje jako kąt 0 stopni północ, i działa skutecznie do kąta 180 stopni.

- Na zakończenie zbieramy wszystkie wyliczone punkty według następującej kolejności: najpierw punkty po lewej stronie prostej w kolejności ich wyznaczania, a następnie punkty po prawej stronie prostej w odwrotnej kolejności.

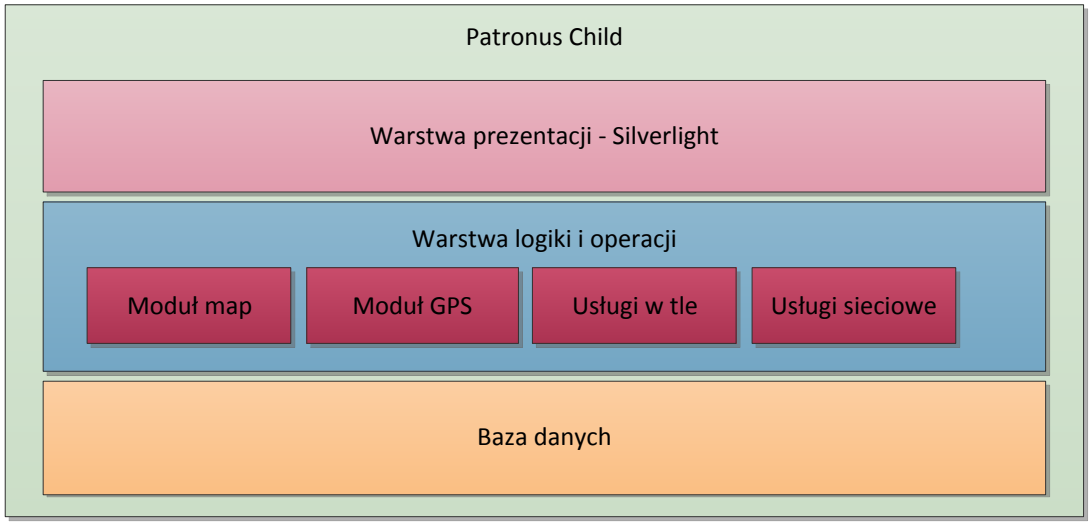
## 6.2 Aplikacja dziecka – Patronus Child

Patronus Child, podobnie jak część serwerowa będzie miał budowę modułową:

- Warstwa bazy danych
- Warstwa logiki i operacji (razem z modułami składowymi)
- Warstwa prezentacji – Silverlight

Głównymi zadaniami tej aplikacji będzie cykliczne sprawdzanie (za uruchomienie tej procedury odpowiadać będzie moduł usług w tle) położenia dziecka za pomocą modułu GPS, i wykorzystując moduł sprawdzania pozycji, zdecydować czy jest ona zgodna z harmonogramem czy też nie.

Oprócz operacji sprawdzania pozycji w tle odbywać się będzie sprawdzanie czy nadszedł już czas synchronizacji danych z serwerem, za którą to odpowiada jak sama nazwa wskazuje moduł synchronizacji.




Rysunek 18 Schemat aplikacji mobilnej dziecka

### 6.2.1 Baza danych

W telefonie będzie funkcjonować również baza danych SQL w wersji na urządzenia mobilne ze schematem identycznym jak baza danych Patronus Serwer, aczkolwiek przechowywany w niej będzie w niej jedynie fragment danych z serwera niezbędny do pracy aplikacji.

### 6.2.2 Warstwa logiki i operacji

Moduł logiki odpowiada za wykonanie i nadzorowanie przebiegu wykonania głównych funkcjonalności aplikacji. Skupia się też na zapewnieniu współpracy wszystkich modułów które się w nim zawierają, a także udostępnia dane do wyświetlenia użytkownikowi, oraz zapewnia obsługę lokalnej bazy danych.

Moduły składające się na tą warstwę to:

- Usługi sieciowe

- Usługi działające w tle
- Moduł GPS
- Moduł synchronizacji
- Moduł sprawdzania pozycji

#### 6.2.2.1 Usługi sieciowe

Komponent ten odpowiada za kontakt z serwerem, za pomocą usług sieciowych. Metody przez niego wykorzystywane są opisane w rozdziale 6.1.2.2.3.

#### 6.2.2.2 Usługi działające w tle

Główny ekran aplikacji dziecka z założenia będzie uruchamiany dość rzadko i większa część jej pracy odbywać się będzie bez udziału użytkownika. Usługi działające w tle mają się budzić cyklicznie i sprawdzać następujące warunki:

1. Czy nadszedł już czas aby dokonać synchronizacji danych ( jeśli tak wyświetlany jest komunikat użytkownikowi). Domyślnie synchronizacja danych powinna odbywać się raz dziennie.
2. Czy nadszedł czas sprawdzenia pozycji (domyślnie co 10 minut). Jeżeli tak to uruchamiany jest moduł GPS w celu pobrania pozycji, a następnie pozycja ta jest sprawdzana pod względem zgodności z harmonogramem przez moduł sprawdzania pozycji. W zależności od odpowiedzi tego modułu możliwe są następujące akcje:
    a. Zgłoszony został alarm nieprawidłowej pozycji – co skutkuje wysłaniem wszystkich danych lokalizacyjnych w telefonie dziecka na serwer, oraz jeśli taka opcja jest aktywna..
    b. Pozycja jest zgodna z harmonogramem – oznacza to, że zapisujemy ją w lokalnej bazie danych, a następnie wykonywane jest sprawdzenie kiedy ostatni raz dane były przesyłane na serwer. Jeśli było to więcej niż 2 godziny temu to wszystkie dane zostają przesyłane na serwer.

    W przypadku wysłania danych na serwer, lokalna baza danych jest czyszczona z wpisów odczytanych z modułu GPS.

#### 6.2.2.3 Moduł GPS

Odpowiada on za użycie wbudowanego w telefon modułu GPS i odczytanie z niego aktualnej pozycji urządzenia, a następnie wyłączenie go by zbędnie nie zużywał energii z baterii.

#### 6.2.2.4 Moduł synchronizacji

Synchronizacja danych ma zapewnić spójność informacji przechowywanych na serwerze i na telefonach użytkowników. Odbywa się ona niemal identycznie na telefonie dziecka i rodzica, aczkolwiek przesyłane są tym aplikacjom rożne dane.

Za uruchomienie procesu synchronizacji odpowiada użytkownik. Oprócz tego wyświetlane są też przypomnienia przez usługi działające w tle, gdy minie termin kolejnej synchronizacji danych.

Sam przebieg procesu synchronizacji danych został opisany w rozdziale *6.1.2.1. Moduł synchronizacji*.

#### 6.2.2.5 Moduł sprawdzania pozycji

Proces sprawdzania pozycji jest uruchamiany przez usługę działającą w tle po odczytaniu pozycji z odbiornika GPS. Następnie ta pozycja jest porównywana przez aktualnie opisywany moduł pod względem zgodności z harmonogramem:

1. Na podstawie godziny odczytu pozycji wyszukiwany jest odpowiedni wpis w harmonogramie. Jeśli takiego nie ma to znaczy że pozycja jest poprawna.
2. Gdy znaleziono więcej niż jeden wpis to brana jest pod uwagę suma obszarów przewidzianych na tą godzinę.
3. Każdy obszar opisuje albo miejsce (koło), albo drogę (łamana wraz obszarem wokół niej), dlatego do sprawdzenia pozycji wybierany jest jeden z dwóch poniższych algorytmów.

##### 6.2.2.5.1 Miejsce

Pierwszy przypadek jest dużo prostszy i sprowadza się do wyznaczenia odległości pomiędzy aktualną pozycją dziecka, a środkiem okręgu w jakim powinno się znajdować, a następnie porównanie tej odległości z promieniem obszaru. Jeśli wyliczona odległość będzie mniejsza, bądź równa, znaczy to, że dziecko znajduje się w obszarze w jakim powinno się znajdować.

Wyznaczanie odległości między dwoma punktami odbywa się według algorytmu:

**Stałe:**

***earthRadius*** – promień Ziemi w kilometrach – 6376 km.

**Zmienne:**

***lat1, lng1, lat2, lng2*** – zmienne określające odpowiednio szerokość (latitude) i długość (longitude) geograficzną, wyrażoną w radianach, pierwszego i drugiego punktu

***Δlat, Δlng*** – różnica pomiędzy odpowiednio szerokością i długością geograficzną dwóch punktów, wyrażona w radianach

**Wynik:**

***distance*** – podana w kilometrach odległość pomiędzy współrzędnymi

**Obliczenia:**

$$\boldsymbol{a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat1) \cdot \cos(lat2) \cdot \sin^2\left(\frac{\Delta lng}{2}\right)}$$

$$\boldsymbol{c = 2 \cdot \tan^{-1}\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right)}$$

$$\boldsymbol{distance = c \cdot earthRadius}$$

##### 6.2.2.5.2 Droga

Pozostaje jeszcze przypadek drugi, czyli sprawdzanie czy aktualna pozycja dziecka znajduje się w obszarze należącym do drogi, dokładniej czy znajduje się wewnątrz wielokąta o zadanej szerokości, który otacza łamaną oznaczającą drogę.

Podejścia jakie można zastosować w tym przypadku są dwa:

1. Jako daną wejściową potraktować informację o drodze przechowywanej w bazie danych i sprawdzić kolejno dla każdego odcinka drogi jego odległość od prostej, która przechodzi przed odcinek. Pozostają tutaj do rozwiązania jeszcze dwa problemy, mianowicie sprawdzenie czy rzut badanego punktu leży na rozpatrywanym odcinku, oraz, jeśli nie leży to sprawdzenie czy mieści się w granicach określonych przez szerokość drogi przy końcach odcinka.
2. Jako daną wejściową potraktować zbiór punktów opisujących obszar wokół drogi, a następnie sprawdzić czy badany punkt znajduje się wewnątrz tego wielokąta, stosując np. algorytm zliczający ilość przecięć półprostej poprowadzonej z badanego punktu z krawędziami wielokąta.

Oba zastosowane algorytmy dadzą ten sam wynik, zatem kryterium wyboru jednego z nich jakie tu zastosowałem uwzględnia fakt, że operacje te będą wykonywane na urządzeniu mobilnym, które chcemy jak najmniej eksploatować (po pierwsze wykonuje wolniej obliczenia, po drugie korzysta przy tym z baterii).

Zastosowanie algorytmu nr 2 wydaje się prostsze, pod warunkiem, że posiadamy zbiór punktów tworzących wielokąt wokół drogi. Bez tego zbioru punkty te musimy wyznaczyć, co znacznie wydłuży czas sprawdzania. Użycie tej metody byłoby opłacalne, gdybyśmy w bazie danych przechowywali dane o wspomnianym wielokącie. Jednak telefon dziecka zgodnie z założeniami nie będzie wyświetlał na mapie zdefiniowanych planów dnia, zatem też nie będzie wyznaczał obszaru wokół drogi.

Zatem metodą wykorzystywaną na telefonie dziecka do sprawdzania pozycji będzie algorytm numer 1.

### 6.2.3 Warstwa prezentacji

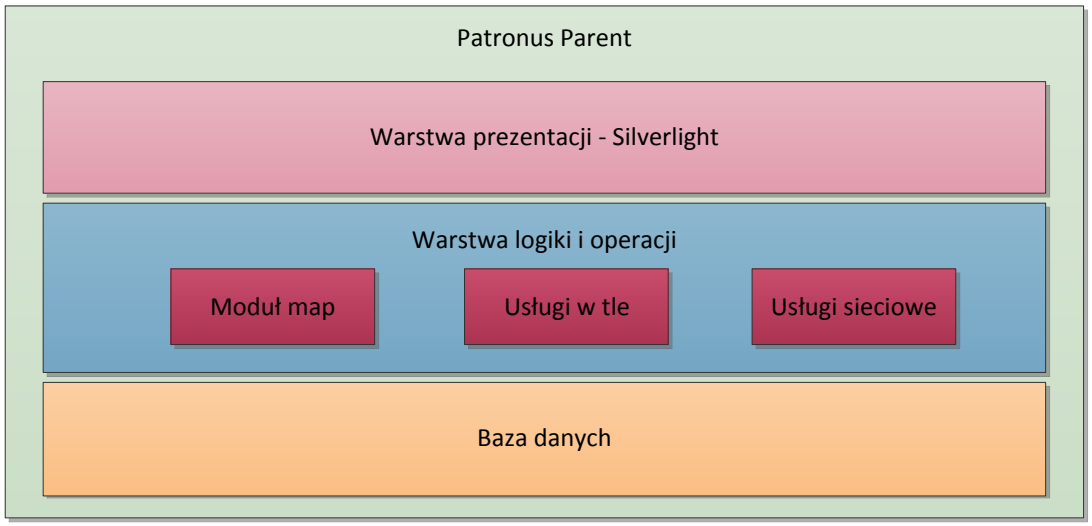
Aplikacja prezentująca dane na telefonie dziecka będzie miała ograniczoną funkcjonalność, głównie służącą do prezentacji obecnego stanu przesłanych danych i przeglądania harmonogramu.

Realizowana będzie jako aplikacja wielostronna z następującymi stronami:

1. Ekran powitalny – zawierać będzie informacje o profilu dziecka na który jesteśmy zalogowani, datę ostatniej synchronizacji danych, godzinę ostatniego przesłania pozycji na serwer, oraz etapy synchronizacji danych gdy proces ten zostanie uruchomiony.
2. Ekran logowania – wyświetlany przy pierwszym uruchomieniu aplikacji,
3. Plan dnia – ekran pozwalający na przeglądania harmonogramu dnia na dzień dzisiejszy.

## 6.3 Aplikacja rodzica – Patronus Parent

Zadaniem aplikacji rodzica (monitorującej), jest cykliczne odpytywanie serwera o zmiany w położeniu geograficznym dzieci i powiadamianie rodzica, w przypadku gdy dziecko znalazło się poza obszarem w jakim powinno być.




Rysunek 19 Schemat aplikacji monitorującej

Oprócz tego podobnie jak w aplikacji dziecka udostępniona jest tu funkcjonalności synchronizacji danych, oraz bardziej rozbudowana warstwa prezentacji, która umożliwia m.in. prezentację danych na mapie.

Aplikacja ta składa się z poniższych warstw:

- Warstwa bazy danych
- Warstwa logiki i operacji
- Warstwa prezentacji

### 6.3.1 Baza danych

Analogicznie jak w aplikacji Patronus Child, Patronus Parent posiadał będzie lokalną bazę danych w wersji lekkiej przeznaczonej dla urządzeń mobilnych.

### 6.3.2 Warstwa logiki i operacji

Moduł ten odpowiada w aplikacji za następujące rzeczy:

1. Zarządzanie prezentowanymi użytkownikowi danymi (przechowywanie ich w pamięci operacyjnej, pobieranie z bazy danych, wyznaczanie punktów do wyświetlenia na mapie)
2. Pracę usług w tle.
3. Pracę modułu synchronizacji.
4. Odpytywanie o zmiany w położeniu dzieci, oraz informowanie rodzica o alarmach.

Składają się na niego następujące moduły:

- Usługi w tle
- Moduł synchronizacji
- Moduł sprawdzania danych

#### *6.3.2.1 Usługi w tle*

Podczas pracy aplikacji i po jej zamknięciu w tle mają odbywać się poniższe operacje:

1. Sprawdzenie czy nadszedł czas synchronizacji danych i jeśli tak to wyświetlenie użytkownikowi komunikatu o tym.

2. Sprawdzenie czy nadszedł czas sprawdzenia danych o pozycjach dzieci na serwerze (domyślnie co 15 minut) i jeżeli tak to uruchamiany jest moduł sprawdzania danych.

#### 6.3.2.2 Moduł synchronizacji

Analogicznie jak w aplikacji dziecka (rozdział 6.2.1.4. ).

#### 6.3.2.3 Moduł sprawdzania danych

Sprawdzanie danych jest uruchamiane przez usługę działającą w tle, po czym wykonywane jest:

1. Wywołanie usługi sieciowej w celu pobrania danych z serwera.
2. Jeżeli serwer zwróci dane to są one zapisywane w lokalnej bazie danych.
3. Jeżeli wśród zwróconych danych znajduje się informacja o alarmie to jest wyświetlany odpowiedni komunikat użytkownikowi.
4. Jako termin ostatniego sprawdzenia danych oznaczana jest godzina rozpoczęcia całej procedury.

### 6.3.3 Warstwa prezentacji

Aplikacja monitorująca składa się z wielu stron, na których prezentowane są konkretne informacje:

1. Logowanie do systemu – strona wyświetlana po pierwszym uruchomieniu aplikacji.
2. Strona główna – prezentowane tutaj są informacje na temat aktualnie zalogowanego użytkownika, terminu ostatniej synchronizacji danych, oraz godziny ostatniego pobrania danych o położeniu dzieci z serwera.
3. Dzieci – lista profili dzieci które obecny użytkownik monitoruje, wraz z informacjami o ich ostatniej aktywności.
4. Plan dnia – po przejściu na ten ekran po wybraniu profilu dziecka, wyświetlane są harmonogramy i wpisy harmonogramów dla zadanego profilu.
5. Plan na dziś – po przejściu na tą stronę z ekranu Plan dnia, wyświetlana jest lista zaplanowanych zajęć na dzisiejszy dzień.
6. Przebieg dnia – po wybraniu profilu dziecka i przejściu na tą stronę wyświetlone zostaną zarejestrowane za dzisiejszy dzień zmiany pozycji dzieci wraz ze specjalnym oznaczeniem alarmów.
7. Mapa – wyświetlane tutaj są dane z takich stron jak Plan dnia, Plan na dziś, Przebieg dnia w widoku mapy

# 7 Implementacja

Przechodząc do fazy implementacji systemu, chciałbym zaznaczyć, że będzie to jedynie przykładowa implementacja, nie pokrywająca wszystkich funkcjonalności, ale zapewniające te najważniejsze i pozwalająca na swobodną pracę systemu. Zgodnie z założeniami projektu, wykonanie systemu nie jest ograniczone do żadnej konkretnej technologii, jednak mogą występować pewne różnice w działaniu systemu uzależnione od tego wyboru. Dostępne na rynku technologie różnią się między sobą na tyle, że często wykonanie czegoś co jest możliwe w jednej, w drugiej jest mocno utrudnione, lub wręcz niemożliwe i na odwrót. Czasem też w trakcie wykonywania aplikacji może okazać się, że wybrana technologia nie pozwala na pełną implementację niektórych zaprojektowanych funkcjonalności, co powoduje naturalny problem, co zrobić w takim przypadku? W dalszej części pracy będzie opisanych kilka takich sytuacji i sposób radzenia sobie z nimi.

Chciałbym pokazać tutaj implementację używającą serwera internetowego do komunikacji, czyli ograniczyć się do trybu pracy systemu internetowego (bez korzystania z wiadomości SMS). Częściowo podyktowane jest to faktem, że jako platformę dla urządzeń mobilnych wybrałem Windows Phone (o czym będzie niżej), ale też tym, że będzie to w mojej opinii dość przyszłościowe rozwiązanie, gdyż nowoczesne smartfony i związane z nimi oferty operatorów sieci komórkowych pozwalają na nieograniczony dostęp do Internetu.

## 7.1 Środowisko pracy

Moim wyborem dla części serwerowej jest użycie technologii .NET 4.0 [10] i Silverlight [11]. Aplikacja rodzica będzie dedykowana na urządzenie z systemem Windows Phone, podobnie jak aplikacja dziecka. Wybór Windows Phone był podyktowany tym, że platforma ta zyskuje coraz większe uznanie na rynku, dlatego postanowiłem jej się przyjrzeć z bliska. Aplikacja na Windows Phone będzie używała technologii Silverlight (druga z możliwych technologii mianowicie XNA przeznaczona jest do tworzenia gier na smartfony, stąd prosta przyczyna dlaczego nie została tu wybrana).

Do pracy całego systemu potrzebny jest również serwer bazy danych, używał będę SQL Server 2008 R2 Express, a do zarządzania nim Management Studio 2008 Express. Natomiast główna część projektu będzie tworzona pod kontrolą Visual Studio 2010 Proffesional, z doinstalowanym darmowym środowiskiem developerskim, pozwalającym tworzyć aplikacje na Windows Phone, czyli Visual Studio Express for Phone. Ogólnie cała przykładowa implementacja korzystać będzie głównie z oprogramowania firmy Microsoft. Visual Studio jest wielkim kompleksowym narzędziem, pozwalającym wygodnie i prosto tworzyć aplikacje, do tego używanym przeze mnie językiem programowania będzie C#, którym zainteresowałem się ze względu na stopień jego dopracowania. Dodatkowo w aplikacjach opartych o Silverlight tworzenie graficznego interfejsu odbywa się w rozszerzonym przez Microsoft języku znaczników o nazwie XAML.

### 7.1.1 Repozytorium kodu

Rozpoczęcie prac z kodem nie może się odbyć bez użycia repozytorium kodu. Nie będę tutaj przytaczał wszystkich argumentów za tym przemawiających, ale chciałbym wspomnieć te główne, czyli po pierwsze repozytorium pełni swego rodzaju kopię zapasową wytwarzanego kodu, po drugie przechowuje historię zmian w nim (co pozwala przywrócić poprzednią działającą wersje, gdy po wprowadzonych zmianach nastąpią problemy), oraz po trzecie, według mnie korzystanie z repozytorium daje większą kontrolę nad tym co tworzymy, oraz zmusza nas do lepszej organizacji pracy.

Oprogramowanie o którym wspominałem do tej pory było dziełem Microsoftu, aczkolwiek w sprawie wersjonowania kodu postanowiłem nie używać tutaj, co prawda wspieranego przez Visual Studio Team Foundation Serwera, gdyż ze względu na to, że system monitoringu będzie tworzony przez jedną osobę, oraz że będzie to raczej mały projekt, to użycie tak 'zasobnożernego' i 'dużego' rozwiązania mija się z celem. Wybrałem zamiast tego 'lżejsze' (dla pamięci operacyjnej) rozwiązanie jakim jest Visual SVN Server [12]. Przekonała mnie do niego prosta instalacja i konfiguracja z poziomu graficznego interfejsu.

Mając już wybrany i zainstalowany serwer kontroli wersji przyszła kolej na wybranie klienta. Z uwagi na fakt, że większość czasu będę spędzał w Visual Studio, zainstalowałem do niego dodatek w postaci Ankh SVN for Visual Studio, który integruje się ze środowiskiem pracy, oraz sygnalizuje zmiany w plikach w widoku rozwiązania (Solution Explorer). By na dobre rozpocząć pracę, konieczne jeszcze było zmienieni ustawień używanej przez VS 2010 wtyczki na wspomniany Ankh SVN, gdyż domyślnie wspierane był oczywiście Team Foundation Server.

### 7.1.2 Użyte technologie

Zanim przejdę dalej do przedstawienia z jakich projektów będzie składało się Rozwiązanie (Solution – nadrzędny 'byt' w Visual Studio, w który organizowane są projekty), krótko chciałbym przedstawić z jakich technologii będę korzystał podczas pracy. Częściowo wspomniałem o tym we wstępie do rozdziału, ale chciałbym ten temat bardziej przybliżyć, nim opowiem w jakiś sposób każda z tych technologii będzie używana.

***ADO.NET Entity Framework*** [13] – zacznę od niego, przyjmując konwencję omawiania technologii od warstwy danych do warstwy prezentacji. Po utworzeniu bazy danychpraktycznym rozwiązaniem jest by korzystanie z niej z poziomu kodu było proste. Wspomniana biblioteka ułatwia to zadanie, mapując tabele z bazy danych na klasy encji w projekcie. Dzięki czemu będzie można korzystać z nich jak ze zwyczajnych obiektów. Co więcej Entity Framework udostępnia klasę, która reprezentuje kontekst bazy danych, czyli z poziomu obiektowego pozwala nam pobierać, dodawać, modyfikować oraz usuwać dane z bazy, bez bezpośredniego kontaktu z nią. Praktycznym rozwiązaniem jest tu odwzorowanie relacji pomiędzy obiektami za pomocą tzw.właściwości nawigujących (ang. Navigation Properties), tworzą one z obu stron przykładowo relacji jeden-do-wielu odpowiednie pola, poprzez które z jednej strony mamy dostęp bezpośrednio do obiektu nadrzędnego, a z drugiej do kolekcji obiektów podrzędnych. Ciekawie działa tu również obsługa relacji wiele-do-wielu, nie mapując do poziomu obiektu tabeli łączącej, a jedynie tworząc pola nawigacyjne do kolekcji obiektów z drugiego końca relacji. Ostatnim udogodnieniem o jakim chciałbym tu wspomnieć, jest mechanizm śledzenia zmian, który gdy pobierzemy z kontekstu obiekt (co odpowiada wczytaniu rekordu z konkretnej tabeli z bazy), to wykonywane na nim operacje będą śledzone przez kontekst, przez co zapisanie zmian w bazie sprowadza się na prostym wywołaniu metody ***SaveChanges()*** na kontekście.

***Windows Communication Foundation*** (WCF) [14] – jest to technologia zorientowana na serwisy, czyli usługi. Zgodnie z założeniami serwer systemu musi udostępniać różnego rodzaju serwisy zarówno aplikacji konfigurującej system (dostępnej przez przeglądarkę internetową), jak i urządzeniom monitorowanym i monitorującym. Wspomniane serwisy będą udostępniane na serwerze. Sam zaś serwer będzie udostępniany np. za pomocą usługi IIS.

***Silverlight*** [11] – technologia, której będę używał do utworzenia aplikacji internetowej, oraz mobilnych. Można powiedzieć, że jest ona odpowiedzią Microsoftu na dominację w Internecie aplikacji Flash i Java. Aplikacje oparte o Silverlight do utworzenia graficznego interfejsu

wykorzystują XAML, który daje większą kontrolę i przejrzystość nad tym co tworzymy (przynajmniej w porównaniu do aplikacji Windows Forms), oczywiście stosowana tutaj jest też technika ***code behind***, czyli kodu 'ukrytego' za interfejsem graficznym, który odpowiada za jego logikę i działanie.

***Mapy Bing*** – interfejs programistyczny pozwalający na korzystanie z map. Przygotowanych jest wiele środowisk deweloperskich (SDK) na różne platformy (Silverlight, Windows Phone, Android, iOS, itp.).

***SQL Server Compact Toolbox*** – bardzo wygodne narzędzie, pozwalające z łatwością pracować z lokalną bazą danych na urządzeniach mobilnych. Po pierwsze pozwala nam utworzyć skrypt tworzący kompaktową bazę danych na podstawie tabel w bazie SQL Server Express R2, a następnie dodanie klas encji bazy danych Compact, oraz kontekstu dostępu do niej do projektu działającego na urządzeniu z systemem Windows Phone 7.

### 7.1.3 Dodatki

Dodatki jak sama nazwa wskazuje nie są niezbędne do pracy, aczkolwiek często ją ułatwiają, lub użyję takiego terminu 'umilają' pracę, a zwłaszcza modyfikacje kodu, którego znaczenia już częściowo zapomnieliśmy. Mam tutaj na myśli dodatek do Visual Studio o nazwie Ghost Doc, który w przemyślany i prosty sposób tworzy komentarze do klas, metod, własności, stałych itp. Komentarze te tworzone są w formacie XML, do którego na podstawie nazwy np. metody i nazw parametrów jej wywołania tworzony jest odpowiedni komentarz, z uzupełnieniem opisów. Co prawda zwykle warto poświęcić chwilę dla opisania jak działa cała metoda, oraz chociaż w dwóch słowach opisać parametry, niż zdawać się na 'domyślność' dodatku.

Przy okazji tej chciałbym wspomnieć o konwencji nazw jaka będzie stosowana w kodzie. W kilku punktach:

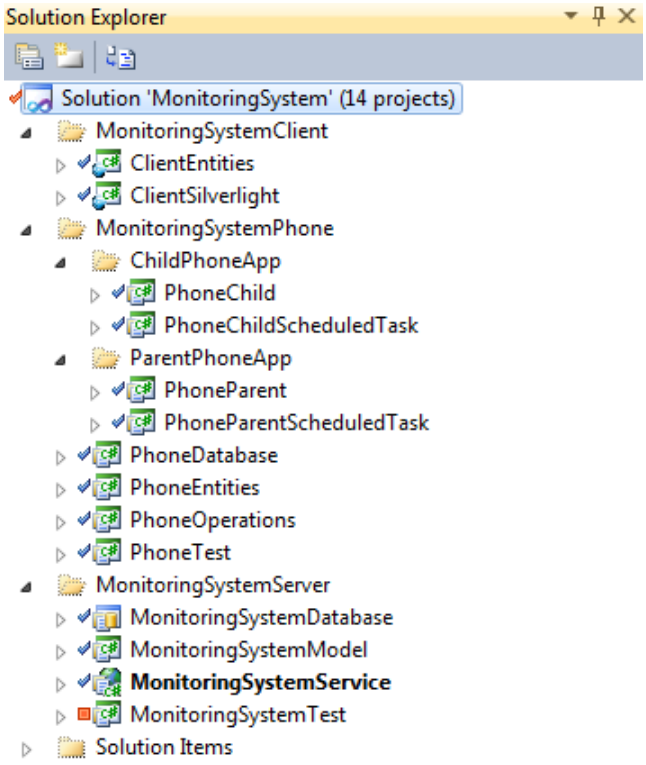
1. Wszystkie nazwy klas, metod, zmiennych itp. Będą w języku angielskim, stosując notację wielbłądzią (przykład nazwy metody : ***MethodName***())
2. Nazwy klas, metod, własności i stałych pisane będą z dużej litery, natomiast nazwy zmiennych z małej.
3. Wyjątkiem są nazwy encji z bazy danych: są to nazwy w języku polskim, oraz pisane z małej litery, a do oddzielania słów używany jest znak '_'. Np. ***nazwa_encji***.

### 7.1.4 Podział na projekty

Organizację pracy postanowiłem podzielić na 3 główne części: część serwerową, część kliencką (aplikacja dostępne przez przeglądarkę internetową), oraz część mobilną, która z kolei składa się z aplikacji rodzica i dziecka. Całość podziału widoczna jest w widoku rozwiązania poniżej.

1. MonitoringSystemServer – czyli całość aplikacji serwerowej – Patronus Serwer
    a. MonitoringSystemDatabase – projekt bazy danych zgodnie z powstałą wcześniej koncepcją, zawierający skrypty tworzące tabele, klucze główne i obce, a także przykładowe dane testowe. Zrezygnowałem z wprowadzania do bazy logiki, czyli m.in. sprawdzania poprawności danych, ponieważ w przypadku błędnie wprowadzonych danych przez użytkownika, baza danych rzuciłaby wyjątek, który należało by obsłużyć, gdyż samo jego wyświetlenie użytkownikowi jest akcją raczej nie przychylnie przyjmowaną. Dlatego też warstwy wyższe będą miały za zadania sprawdzenie danych przed ich zapisaniem do bazy.

b. MonitoringSystemModel – projekt wykorzystujący Entity Framework, do obiektowego zmapowania tabel bazy danych, czego wynikiem jest utworzenie klas im odpowiadających, a także klasy kontekstu, która pozwala na wykonywanie operacji na bazie danych.
c. MonitoringSystemService – ten projekt pełni dwie role, po pierwsze stanowi serwer udostępniający serwisy do komunikacji z nim aplikacji klienckiej (przeglądarkowej), oraz aplikacjom mobilnym. Drugą pełnioną przez niego rolą jest hostowanie aplikacji klienckiej, przez co stanie się ona dostępna użytkownikowi właśnie przez ten serwer. Projekt odpowiada za główną część logiki systemu, zarządza przepływem danych pomiędzy komponentami, oraz odpowiada za kontakt z bazą danych.
d. MonitoringSystemTest – testy funkcjonalności dotyczących części serwerowej, głównie operacji wykonywanych przez usługi sieciowe




Rysunek 20 Podział systemu na projekty

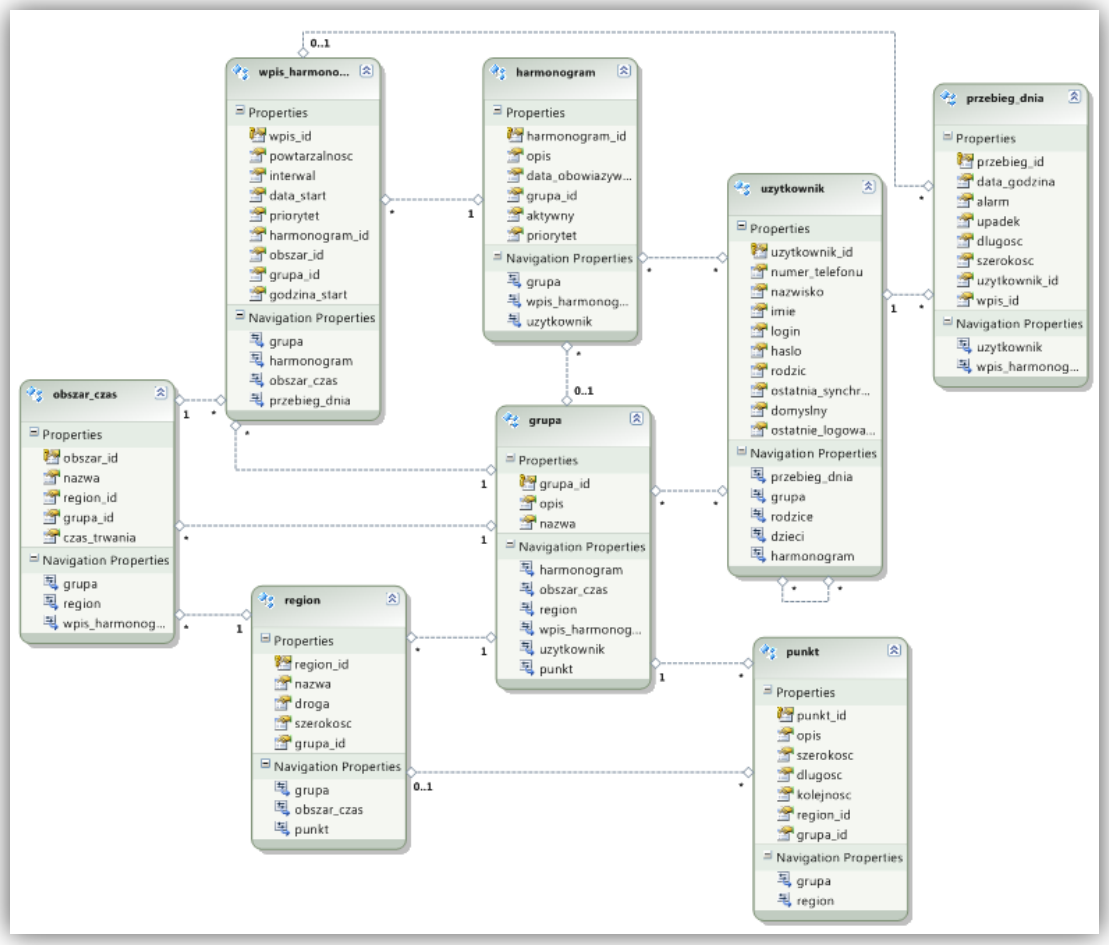
2. MonitoringSystemClient – część systemu udostępniana użytkownikowi przez przeglądarkę internetową (Patronus Viewer).
   a. ClientEntities – wyodrębniony projekt, na potrzeby wykorzystywania przez aplikację kliencką utworzonych przez Entity Framework encji – klas powstałych ze zmapowania tabel z bazy danych. Klient tworzony jest w technologii Silverlight, przez co nie możemy bezpośrednio skorzystać z odwołania do projektu MonitoringSystemModel (projekt .NET 4.0) i potrzebna jest warstwa pośrednia.
   b. ClientSilverlight – aplikacja przeglądarkowa pozwalająca użytkownikowi na zarządzanie systemem monitoringu – tworzenie profili innych użytkowników, planowanie harmonogramów, definiowanie obszarów, oraz przeglądanie historii pracy systemu.

3. MonitoringSystemPhone – część mobilna systemu.
    a. PhoneEntities – podobnie jak w przypadku ClientEntities, jest to projekt stanowiący warstwę pośrednią i pozwalający na używanie obiektów encji bazy danych w aplikacjach Windows Phone 7.
    b. PhoneDatabase – biblioteka zawierająca klasy encji z lokalnej bazy danych (SQL Server Compact), a także klasę kontekstu dostępu do tej bazy, oraz maper, który służy do zamiany obiektów z klas bazy danych wykorzystywanych po stronie serwera, na obiekty lokalne, oraz na odwrót.
    c. PhoneOperations – projekt służący do kontaktu aplikacji mobilnych z lokalną bazą danych, oraz wywołujący usługi sieciowe. Odpowiada za główną część logiki aplikacji mobilnych. Tutaj wykonywane są operacje sprawdzania pozycji, wyznaczania obszarów do wyświetlenia na mapie, oraz proces synchronizacji danych z serwerem.
    d. PhoneParentApp – folder skupiający projekty wyłącznie działające na telefonie rodzica
        i. PhoneParent (Patronus Parent) – aplikacja monitorująca (rodzica), która pozwala na sprawdzanie aktualnej pozycji dziecka, przejrzenie jego planu dnia, oraz która powiadamia użytkownika w przypadku gdy dziecko znajdzie się w nieodpowiednim miejscu, aby rodzic mógł szybko zareagować.
        ii. PhoneParentScheduledTask – zaplanowane zdarzenie wywoływane cyklicznie przez Scheduled Action Service, które sprawdza konieczność synchronizacji, oraz odpowiada za pobieranie z serwera informacji o pozycjach dzieci, a także zgłaszanie alarmów.
    e. PhoneChildApp – folder skupiający projekty wyłącznie działające na telefonie dziecka.
        i. i PhoneChildScheduledTask – odpowiadają za uruchamiane przez tzw. serwis zaplanowanych akcji (Scheduled Action Service) zdarzenia, czyli wykonanie wszystkich zaplanowanych akcji przez telefon właściciela, jak m.in. odpytanie serwera o położenie dzieci (w przypadku telefonów rodziców), lub przesłanie tych informacji (telefony dzieci).
        ii. PhoneChild (Patronus Child) – aplikacja monitorowana (dziecka), a w zasadzie jej konfigurator, uruchamiana jest podczas instalacji aplikacji na telefonie dziecka, oraz podczas synchronizacji danych z serwerem.
    f. PhoneTest – projekt służący do testowania funkcjonalności działających na aplikacjach mobilnych

## 7.2 Patronus Serwer

Serwer jak już wspomniano wcześniej składa się z trzech projektów: Bazy danych, Modelu bazy danych i Usług sieciowych. Na projekt bazy danych składają się głównie skrypty ją tworzące zatem nie ma potrzeby wyjaśniać więcej szczegółów. Baza danych jest utworzona zgodnie ze schematem opisanym w dziale architektury. Inaczej jest w przypadku pozostałych dwóch projektów, w których konieczne było zastosowanie ciekawych rozwiązań podczas implementacji.

### 7.2.1 Model bazy danych




**Rysunek 21 Model bazy danych utworzony za pomocą Entity Framework**

Głównym elementem modelu bazy danych jest plik ***Model.edmx***, tworzony za pomocą EntityFramework na podstawie utworzonej bazy danych (jak w projekcie). W dalszej części niezbędne było utworzenie plików z kodem klas, które będą w systemie odpowiadać tabelom bazy danych. Posłużyło temu narzędzie o nazwie ***TextTemplatingFileGenerator***, które wygenerowało kod na podstawie istniejących we Framework'u szablonów.

Do zgłębienia wiedzy na temat prawidłowego użycia wspomnianych przeze mnie narzędzi, oraz w następnej kolejności możliwości współpracy całego systemu z bazą danych, poprzez usługi sieciowe i wspomniany EntityFramework bardzo przydatny był artykuł zatytułowany „Entity Framework – aplikacja trójwarstwowa" [15].

### 7.2.2 Warstwa logiki

Warstwa logiki realizowana jest przez moduł ***MonitoringSystemService***.

Możemy go podzielić na 3 części:

1. Komunikacja z bazą danych za pomocą klasy kontekstu dostępu do bazy danych.
2. Serwisy pozwalające na dostęp do informacji z bazy danych:
    a. PhoneService – serwis Windows Communication Foundation pozwalający na komunikację urządzeniom mobilnym.

b. Service – serwis przeznaczony do komunikacji z aplikacjami tworzonymi w technologii Silverlight – tak jak to ma miejsce w aplikacji zarządzającej.

3. Udostępnianie (hosting) aplikacji zarządzającej.

#### 7.2.2.1 Usługi sieciowe

Najważniejszą częścią biblioteki realizującej warstwę logiki są usługi sieciowe. Do użycia ich konieczne jest dodanie do projektu dwóch plików konfiguracyjnych, które określają politykę dostępu do usług sieciowych spoza własnej domeny (opcja ***allow-from*** w pierwszym pliku i ***allow-http-request-headers-from*** w drugim pliku akceptuje połączenia przychodzące z każdej domeny):

```
<?xml version="1.0" encoding="utf-8" ?>
<access-policy>
  <cross-domain-access>
    <policy>
      <allow-from http-request-headers="*">
        <domain uri="*"/>
      </allow-from>
      <grant-to>
        <resource include-subpaths="true" path="/"/>
      </grant-to>
    </policy>
  </cross-domain-access>
</access-policy>
```

Rysunek 22 clientaccesspolicy.xml

Oraz

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
       "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-http-request-headers-from domain="*" headers="*"/>
</cross-domain-policy>
```

Rysunek 23 crossdomain.xml

Są to, zgodnie z zaleceniami MSDN [16], pliki zapewniające możliwość działania serwisów aplikacjom napisanym w Sliverlight 4 lub wyższym, gdzie sprawdzanie komunikacji między różnymi domenami jest dość restrykcyjnie i musimy poinformować system, że taka komunikacja jest przez nas pożądana, gdyż klienci serwisów będą się łączyli z serwerem z różnych domen.

Bardzo ważnym aspektem udostępniania usług sieciowych jest zapewnienie bezpiecznego kanału przesyłania danych, używającego SSL. Konfiguracja serwera udostępniającego usługi sieciowe przez kanał zaszyfrowany jest opisana w artykule [17].

Istnieją również dwa przypadki gdy do klienta zamiast odpowiedzi będzie wysyłany komunikat o błędzie:

- Brak autoryzacji (sesja wygasła, lub jej nie nawiązano jeszcze)
- Dane na serwerze zostały zmienione (wyjątek zgłaszany przez bazę danych przy próbie modyfikacji danych które już zostały zmienione/usunięte, lub o naruszeniu więzów integralności)

### 7.2.3 Patronus Viewer

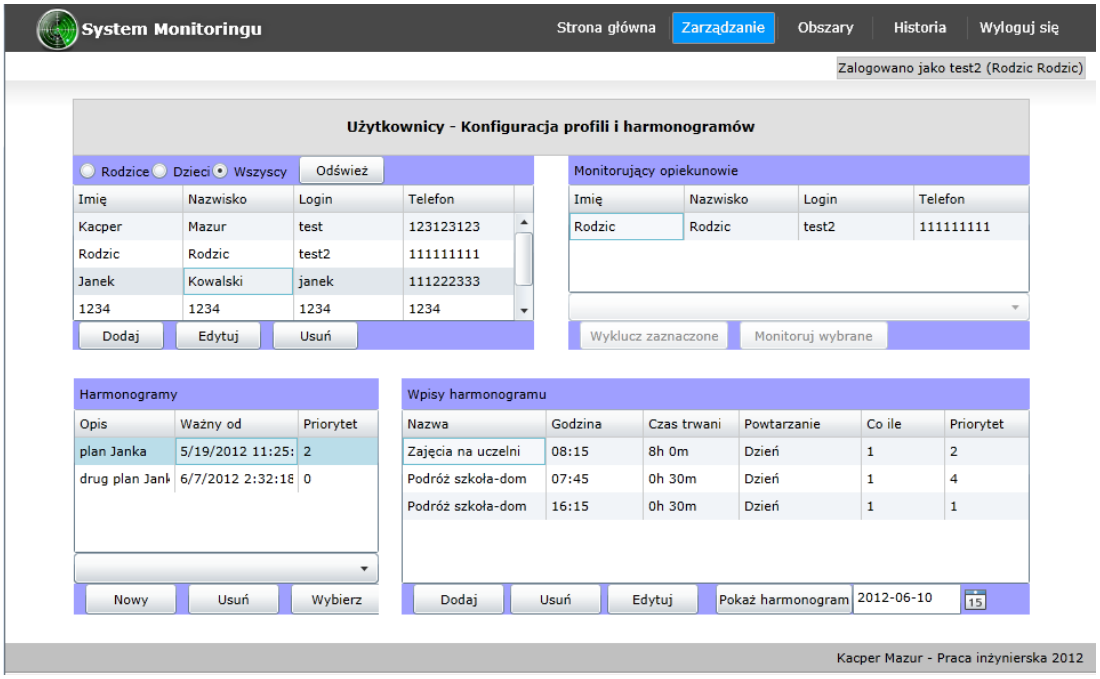
Aplikacja ta jest aplikacją kliencką, która komunikuje się z serwerem. Do jej implementacji, jak wspomniano w kilku miejscach wyżej użyta została technologia Silverlight.

Tworzenie aplikacji o nią opartych jest bardzo podobne do tworzenia aplikacji Windows Presentation Foundation. Podobnie do opisu interfejsu graficznego używany jest rozszerzony język znaczników – XAML.

Cała aplikacja jest to oparta o szablon o nazwie ***Silverlight Navigation Application.*** W ramach niego tworzona jest jedna strona główna (MainPage), w której umieszczane są odpowiednie podstrony (dziedziczące po Page). Możliwe jest również wyświetlanie modalnych okien, które przykrywają stronę główną (dziedziczące po ChilWindow).

Aplikacja składa się z sześciu stron, którymi kolejno są:

- Home.xaml – strona główna, zawierająca powitalne komunikaty
- Login.xaml – strona pozwalająca zalogować się do systemu (przy czym logowanie jest w wersji bardzo uproszczonej)
- Register.xaml – strona rejestracji nowego użytkownika w systemie, i tym samym utworzenie dla niego nowej grupy użytkowników którymi może zarządzać.
    - UsersList.xaml – strona zarządzania użytkownikami i harmonogramami , pozwala na dodawanie nowych profili, decydowanie jakie dzieci mają być monitorowane przez którego rodzica, oraz pozwalająca na tworzenie harmonogramów RegionsList.xaml – strona definiowania regionów i na ich podstawie obszarów
- HistoryList.xaml – strona zezwalająca na przeglądanie historii przebiegów dnia, udostępnia proste filtry, pozwalające wybrać za jaki okres i kogo dotyczące wpisy chcemy zobaczyć




Rysunek 24 Zarządzanie użytkownikami i harmonogramami.

Oprócz tego do pracy aplikacji użyte zostały wspomniane modalne okna, których opis i przeznaczenie znajduje się poniżej:

- ErrorWindow.xaml – ekran wyświetlany gdy wystąpi nieobsłużony wyjątek, szczegóły na jego temat są tutaj wyświetlane (np. podczas problemów komunikacją z serwerem)
- UserDialog.xaml – okno ze szczegółami użytkownika – używane podczas jego dodawania i edycji.
- PlanDialog.xaml – okno wyświetlane podczas dodawania nowego harmonogramu
- RegionDialog.xaml – okno dodawania i podglądu regionu, korzysta z kontrolki Bing Map, do definiowania obszarów (będzie opisane dalej)
- PlaceDialog.xaml – okno definiowania obszaru, który powstaje z wybranego regionu, oraz pozwala na określenie ile czasu dziecko będzie się na nim znajdować
- PlanEntryDialog.xaml – okno odpowiadające pojedynczemu wpisowi w harmonogramie, pozwala na uzupełnienie informacji o wpisie (kiedy i z jaką częstotliwością zdarzenie będzie miało miejsce), oraz jakiego obszaru on dotyczy
- PlanEntryMapDialog.xaml – dla określonej daty wyświetlany jest cały plan dnia (na mapie) na podstawie wszystkich wpisów aktualnie wybranego harmonogramu

#### 7.2.3.1 Kontroler

Pracą całości aplikacji, jak i komunikacją z serwerem zajmuje się klasa ***ClientController***. Korzysta ona z wzorca projektowego Singleton, co zapewnia nam, że dla danego klienta istnieje tylko jeden jej obiekt, przez co likwidujemy problem braku synchronizacji danych po stronie przeglądarki (a konkretnie różnych obiektów kontrolera).

Wspomniany kontroler udostępnia metody do komunikacji z serwerem, oraz używa mechanizmu zdarzeń do powiadamiania okien i stron o otrzymaniu odpowiedzi z serwera.

Do komunikacji z serwerem aplikacja korzysta z tzw. odwołania do usługi (Service Reference), która tworzy na podstawie metod serwisu do którego się odwołuje odpowiadające usługom sieciowym metody asynchroniczne. Co za tym idzie, aby odebrać informacje od serwera musimy dodać do każdej z nich metodę wywoływaną po przyjściu odpowiedzi. Każda z metod serwisu po stronie klienta jest opatrzona końcówką Async, np. ***GetUsersByGroupAsync***, oraz tworzone jest dla niej zdarzenie z końcówką Completed, np. ***GetUsersByGroupComplete***, które wołane jest po uzyskaniu odpowiedzi od serwera.

Do każdej metody sieciowej dodawane są parametry umożliwiające uwierzytelnienie użytkownika gdy trwa sesja (czyli identyfikator użytkownika i klucz sesji). Natomiast gdy sesja wygasła lub nie została jeszcze nawiązana, proces uwierzytelniania składa się z dwóch etapów (zgodnie z założeniami w rozdziale Architektura), czyli zapytania o klucz sesji i odesłania skrótu hasła użytkownika połączonego z kluczem sesji.

Oprócz tego przy każdej odpowiedzi otrzymanej z serwera następuje sprawdzenie czy nie został zakomunikowany błąd, i następnie zostaje on obsłużony (w przypadku wygaśnięcia sesji następuje wylogowanie i wyświetlenie komunikatu użytkownikowi).

W czasie pomiędzy wywołaniem metody usługi sieciowej, a otrzymaniem odpowiedzi kontroler wyświetla okno oczekiwania, informujące użytkownika o tym, że trwa jakaś operacja, inaczej mógłby on być zdziwiony opóźnieniem z jakim wykonywane są niektóre operacje.

Jednak aby zminimalizować czas czekania użytkownika na wykonanie operacji wymagających kontaktu z serwerem, kontroler po pierwszym wczytaniu np. listy użytkowników z serwera, przechowuje ją w pamięci, i odświeża te informacje po wykonaniu jakiejś operacji na nich, lub też na życzenie użytkownika. Ostatni przypadek, który skutkuje przeładowaniem danych to otrzymanie z serwera komunikatu błędu, o tym, że dane które próbujemy zmodyfikować zostały już zmienione lub usunięte. Użytkownik otrzyma stosowny komunikat po którym dane zostaną ponownie wczytane.

#### 7.2.3.2 Mapy Bing

Rozpoczynając pracę z mapami Bing korzystałem z dokumentacji [18] na stronie MSDN, która dość przejrzyście pozwala zapoznać się z podstawami korzystania z nich.

Żeby wykorzystać w praktyce mapy za pomocą odpowiedniego API (w moim przypadku dla Silverlight lub Windows Phone), należy się najpierw zalogować na stronie http://www.bingmapsportal.com/ , w celu wygenerowania klucza, który pozwoli następnie wykorzystywać w pełni mapy. Konieczne jest określenie do jakiego programu będziemy wykorzystywali mapy Bing, oraz wybranie jego typu. Na szczęście możliwy był wybór typu „Education", dzięki czemu bez żadnych przeszkód i dodatkowych opłat mam możliwość pracy z API Bing Maps.

Powodów dlaczego musimy posiadać klucz dla każdej aplikacji korzystającej z map jest prawdopodobnie kilka. Pierwszym jest według mnie bezpieczeństwo, gdyż kontrolki podczas pracy komunikują się z serwerem map, który gdy posiadamy klucz wie, że klient jest zaufany i ryzyko ataków DoS jest bardziej kontrolowane. Drugi powód, prawdopodobnie bardziej oczywisty to względy finansowe. Użycie map w aplikacjach komercyjnych jest płatne, a nie posiadając zakupionego odpowiedniego klucza, mapy nie będą w pełni funkcjonalne. Z dodatkowych powodów warto chociażby wymienić możliwość przeglądania raportów z pracy naszej aplikacji.

Przechodząc do wykorzystania API w praktyce, możemy się już domyśleć, że podstawowym obiektem przez nas wykorzystywanym będzie kontrolka mapy. Bardzo miłym udogodnieniem jest fakt, że kontrolka nie tylko po prostu wyświetla mapę, ale też obsługuje podstawowe zdarzenia z nią związane (np. przybliżanie, zmienianie trybu wyświetlania). Lecz to do czego chciałbym wykorzystać kontrolkę, to wyświetlenie na niej obszarów zdefiniowanych przez użytkownika. Odbywać się to będzie zgodnie z wizją przestawioną podczas projektowania systemu, mianowicie użytkownik będzie musiał na początku pracy z systemem utworzyć regiony (miejsca gdzie dziecko może przebywać, lub drogi jakimi ma się poruszać), a na ich podstawie obszary, czyli też regiony, ale z dodatkowym określeniem czasu trwania pobytu dziecka w tym miejscu. Zabieg ten był celowy, aby możliwe było wielokrotne wykorzystanie regionów.

Organizacja wyświetlania czegokolwiek na kontrolce sprowadza się do utworzenia warstwy nałożonej na mapę, a następnie dodawania do niej obiektów. Udostępnione mamy dwa rodzaje kształtów:

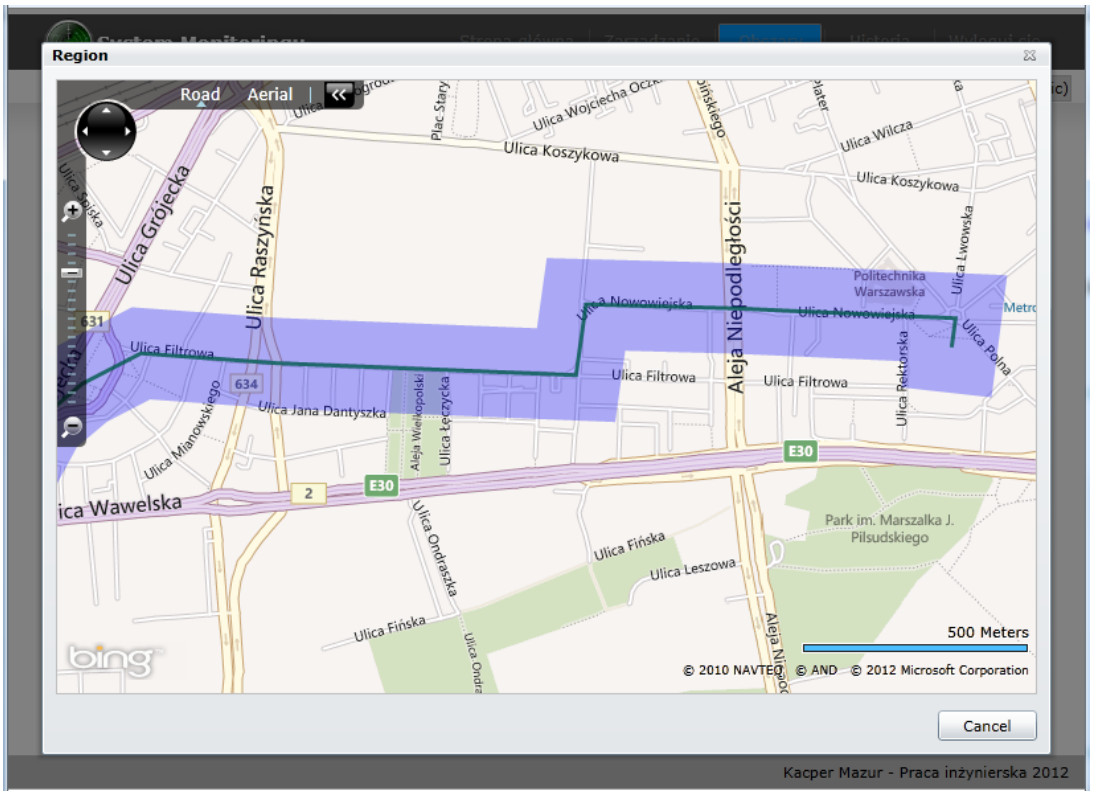
1. Linia łamana (Polyline) – składają się na nią punkty określane parą współrzędnych geograficznych, i ułożone w kolejności tworzą one drogę
2. Wielokąt (Polygon) – podobnie jak łamana składa się z kolejnych par współrzędnych geograficznych, jednak począwszy od trzech punktów, ostatni punkt łączy z pierwszym i zamalowuje środek powstałej figury

Niestety nie mamy możliwości skorzystania z wbudowanego wyświetlania na mapie okręgu o zadanym środku (współrzędne geograficzne), oraz o promieniu np. w metrach. Dlatego zgodnie z algorytmem opisanym w poprzednim rozdziale, koło będzie przybliżane wielokątem.

Możliwe jest również tworzenie tradycyjnych kształtów na mapie, jak elipsy, punkty, prostokąty, jednak mimo, że umieszczając je na warstwie nad mapą używa się do tego współrzędnych geograficznych, to sam rozmiar figur określany jest w pikselach, przez co używanie powiększenia i przybliżenia mapy na te figury nie zadziała.

Chcąc zdefiniować region, użytkownik musi zdecydować czy będzie to droga czy miejsca, oraz podać odpowiednio szerokość, lub promień. Następnie włącza tryb dodawania punktów na mapie, a po zakończeniu dodawania, na podstawie zadanych punktów (lub punktu), tworzona jest odpowiednia figura i wyświetlana jest na mapie zamiast punktów.

Natomiast przy wyświetlaniu regionu, lub obszaru już istniejącego w systemie, mapa zostaje na nim wyśrodkowywana (w przypadku drogi, centrowanie odbywa się na pierwszy punkt na drodze). Kolejną rzeczą przed wyświetleniem, jest dobranie odpowiedniego poziomu przybliżenia, na podstawie tego ile wynosi wysokość wyświetlanej kontrolki mapy (w pikselach), oraz ile wynosi promień/szerokość wyświetlanego obszaru.




Rysunek 25 Przykładowy region w widoku mapy (droga wraz z obszarem dookoła)

## 7.3 Aplikacje mobilne

Zarys pracy dla obu aplikacji na urządzenia mobilne jest bardzo podobny, różnią się one jedynie niektórymi funkcjonalnościami. Zgodnie ze skróconym opisem na początku podrozdziału, dla obu aplikacji mobilnych wyróżnione są trzy biblioteki: PhoneEntities, PhoneDatabase, PhoneOperations.

### 7.3.1 Baza danych

***PhoneEntities*** – biblioteka zawierająca klasy encji wykorzystywane po stronie serwera. Jest potrzebna przy otrzymywaniu informacji z usług sieciowych z serwera, gdyż zwykle będą to informacje z tamtejszej bazy danych.

***PhoneDatabase*** - z uwagi na fakt, że urządzenia mobilne będą potrzebowały składować niektóre informacje otrzymane z serwera, to zgodnie z architekturą zastosowana będzie tutaj tzw. lekka baza danych, a dokładniej baza w wersji SQL Server Compact 3.5. Co prawda najnowszą wersją jest wersja 4.0, ale z uwagi na fakt, że nie jest ona w pełni kompatybilna wstecz, przy próbie pracy z nią niektóre funkcjonalności sprawnie działające w wersji wcześniejszej przestają działać. Dokładniej mam tutaj na myśli wykorzystanie narzędzia o nazwie SQL Server Compact Toolbox [19], dzięki któremu możliwe jest z poziomu Visual Studio zarządzanie wspomnianą kompaktową bazą danych. Jednak dwie funkcjonalności najbardziej przydatne podczas implementacji systemu, są to:

- Generowanie skryptu bazy danych na podstawie już istniejącej bazy – dzięki temu w prosty sposób możliwe było wygenerowanie skryptu na podstawie bazy danych systemu monitoringu istniejącej na SQL Server, w wersji odpowiedniej dla SQL Server Compact, a następnie w prosty sposób utworzenie nowej kompaktowej bazy poprzez uruchomienie skryptu.
- Utworzenie kontekstu dostępu do bazy kompaktowej, oraz wygenerowanie klas encji na podstawie tabel istniejącej bazy kompaktowej.

Podczas pierwszego kontaktu aplikacji mobilnej z bazą danych, konieczne będzie jej utworzenie. Będzie to pusta baza danych, która zostanie uzupełniona podczas pierwszej synchronizacji.

Ważnym faktem pracy z bazą danych będzie to, że obiekty klas wygenerowanych przez Entity Framework po stronie serwera nie będą mapowana automatycznie na obiekty kompaktowej bazy danych. Służyła będzie do tego statyczna klasa mapująca jedne obiekty na drugie (dokładniej interesujące nas pola tych obiektów).

Druga z ważniejszych uwag dotyczy tego że po stronie mobilnej bazy danych zrezygnowałem z przechowywania informacji o relacjach, a tym samym ze skryptu wygenerowanego przez SQL Server Compact Toolbox przed wykonaniem go na bazie danych zostały usunięte wszystkie klucze obce, a także parametr IDENTITY dotyczący kluczy głównych.

Usunięcie kluczy obcych było konieczne, ponieważ podczas synchronizacji kolejne tabele będą czyszczone i zapełniane nowymi danymi, na co nie pozwoliłby więzy integralności nakładane przez klucze obce.

Natomiast wyłączenie automatycznego numerowania kluczy głównych na wszystkich tabelach ma na celu używanie tych samych kluczy głównych jak w bazie danych serwera. Wyjątkiem jest tutaj tabela ***Przebieg_dnia***, gdyż rekordy tej tabeli w przeciwieństwie do wszystkich pozostałych będą tworzone w aplikacji monitorowanej, a nie po stronie serwera. Podczas jednak mapowania ich na klasy działające po stronie serwera, klucz główny nie jest mapowany. Po przesłaniu danych, w bazie danych SQL Server Express dla każdego rekordu zostanie nadany nowy klucz główny.

### 7.3.2 Warstwa logiki

Można powiedzieć, że w pewnym sensie jest to odpowiednik kontrolera zastosowanego w Patronus Viewer. Stanowi on warstwę która pośredniczy między aplikacją mobilną a serwerem, oraz lokalną bazą danych.

Realizowana jest jako Singleton, w którego konstruktorze prywatnym dodana jest obsługa wszystkich odpowiedzi wywołań usług sieciowych.

#### *7.3.2.1 Usługi sieciowe*

Każda usługa sieciowa jest wywoływana asynchronicznie, a następnie po przyjściu odpowiedzi z serwera uruchamiane jest odpowiednie zdarzenie. Metody uruchamiane przez te zdarzenia są właśnie przypisywane konstruktorze prywatnym, aby dla każdej z nich była przypięta jedynie jedna metoda obsługująca.

Dla wygody czytania i analizowana kodu, metody są pogrupowane w tzw. regiony, które pozwalają zwijać zawarte w nich linie kodu w jedną. Za pomocą tego sposobu kod pogrupowany jest w:

- Metody odpowiadające za synchronizację danych z telefonem rodzica
- Metody odpowiadające za synchronizację danych z telefonem dziecka
- Metody dostępu do bazy danych
- Metody wysyłające i odbierające informacje o przebiegu dnia dzieci
- Metody sprawdzające pozycję z harmonogramem
- Metody służące do wyznaczania danych na potrzeby wyświetlania w aplikacji rodzica

Każde otrzymanie odpowiedzi z serwera, przed odczytem wartości zwracanej musi wykonać sprawdzenie czy nie było jakiegoś błędu po drodze, np. serwer jest niedostępny, lub nie mamy po prostu połączenia z Internetem. Wszystkie metody wołane przy otrzymaniu odpowiedzi posiadają dwa parametry: ***sender*** – parametr opisujący obiekt wywołujący metodę, oraz ***e*** – parametr zawierający szczegóły zdarzenia, takie jak obiekty zwracane przez serwer (***Result***), czy informacje o błędach (***Error***).

Przed odczytaniem wartości odpowiedzi z serwera, musimy sprawdzić czy parametr ***e.Error***, zwiera jakąś wartość (jest różny od NULL). Jeśli tak jest to znaczy, że podczas kontaktu z serwerem wystąpił wyjątek, i taką sytuację musimy obsłużyć.

Obsługa tych błędów zależy od sytuacji w jakiej on wystąpił:

- Podczas logowania – wyświetlany jest komunikat o błędzie logowania i ponownej próbie.
- Podczas synchronizacji danych – synchronizacja nie jest oznaczana jako wykonana, nadal będzie konieczne ponowne jej uruchomienie.
- Podczas pobierania informacji o przebiegu dnia – nie jest zmieniana informacja o ostatniej dacie logowania do systemu, i przy następnym pobieraniu danych pobrane zostaną dane od starej daty.
- Podczas wysyłania informacji na serwer – telefon dziecka, co określony interwał czasu, lub przy wystąpieniu alarmu przesyła na serwer informacje o przebiegu dnia, ale gdy wystąpi błąd podczas sprawdzania potwierdzenia z serwera, to nie kasuje tych danych w bazie lokalnej, lecz przy następnej okazji próbuje je ponownie wysłać.

#### 7.3.2.2 Usługi działające w tle

Windows Phone od wersji 7.5 (o nazwie kodowej Mango), pozwala na implementację usług działających w tle(ang. ***Background Agent Tasks*** [20]***)***, oraz udostępnia statyczną klasę Serwisu zaplanowanych akcji (ang.***ScheduledActionService)***, która zarządza wszystkimi zaplanowanymi akcjami (m.in. też służy do planowania alarmów). Serwis zaplanowanych akcji ponadto jest odpowiedzialny za ich wywoływanie. W aktualnej wersji systemu Windows Phone nie mamy możliwości określania godziny wywołania usług działających w tle, decyduje o tym system operacyjny oraz typ usługi. Dla każdej aplikacji Windows Phone możliwe jest utworzenie jednej usługi każdego typu.

Do wyboru mamy dwa typy usług:

- Zadanie cykliczne (ang. Periodic Task) – które jest wywoływane przez system operacyjny co około 30 minut (jak podaje dokumentacja) i jego działanie powinno się zamykać w około 25 sekundach.
- Zadanie wymagające większych zasobów (ang. Resource Intensive Task) – usługa tego typu wywoływana jest rzadziej niż zadanie cykliczne, lecz ma dostęp do większej ilości zasobów telefonu i może działać dłużej (do 10 minut), jednak do wywołania tego zdarzenia muszą być spełnione konkretne warunki:
    - Urządzenie musi być podłączone do zewnętrznego źródła zasilania.
    - Wymagane jest nawiązanie połączenia WiFi, lub z komputerem PC.
    - Poziom naładowania baterii musi być ponad 90%.
    - Ekran telefonu musi być zablokowany.Nie trwa  żadna rozmowa telefoniczna.

By było możliwe korzystanie z powyższych usług, wystarczy skorzystanie z przygotowanego szablonu projektu (ang. Scheduled Agent Task), i implementacja zdarzenia które ma się wykonać po wywołaniu usługi przez system operacyjny. Mamy również tutaj możliwość rozróżnienia co dzieje się w przypadku zadań cyklicznych, a co w przypadku zadań wymagających większych zasobów.

Zgodnie wcześniejszymi wytycznymi, zadania wymagające większych zasobów będą użyte do sprawdzania czy konieczna jest synchronizacja danych (sprawdzenie to nie musi być wykonywane tak często jak pobranie danych z serwera o przebiegu dnia dzieci), oraz jeśli taka synchronizacja jest koniecznie, wyświetli rodzicowi stosowny komunikat, który po kliknięciu uruchomi główne okno aplikacji i pozwoli na rozpoczęcie synchronizowania danych, lub też uruchomi automatycznie proces synchronizacji.

W przeciwnym wypadku, kiedy synchronizacja konieczna nie będzie, zadanie to będzie działało analogicznie do usługi zadania cyklicznego. Mianowicie w kolejnych krokach:

1. Sprawdzi czy minął odpowiedni interwał czasu od ostatniego pobrania danych.
2. Jeśli nie to kończy działanie usługi, a jeśli tak to wykonuje ją dalej.
3. Korzysta z klasy operacji w celu wywołania odpowiedniej usługi sieciowej.
4. Po otrzymaniu odpowiedzi, wprowadza dane do lokalnej bazy danych i jeśli wśród otrzymanych informacji znajdował się alarm, to rodzicowi wyświetlany jest stosowny komunikat.

Poniższy fragment kodu prezentuje wykonywaną przez zdarzenie działające w tle, po uruchomieniu jej przez system operacyjny (metoda ***OnInvoke())***

```
        /// <summary>
        /// Agent obsługujący usługę wołaną przez serwis zaplanowanych usług.
        /// </summary>
        /// <param name="task">
        /// Wywoływana usługa.
        /// </param>
        /// <remarks>
        /// Ta metoda jest wołana przy uruchomieniu Periodic Task,
        /// lub Resource Intensive Task
        /// </remarks>
        protected override void OnInvoke(ScheduledTask task)
        {
            // Rozróżnienie jakie akcje mają być wykonywane przez Periodic Task,
            // a jakie przez Resource Intensive Task
            if (task is PeriodicTask)
                // Sprawdzenie i jeśli to konieczne pobranie
                // informacji z serwera o przebiegu dnia dzieci
                CheckDayEvents();
            else
            {
                // Sprawdzenie czy synchronizacja danych jest konieczna
                if (Operations.Instance.IsSynchNeeded())
                {
                    // jeśli tak to wyświetlamy komunikat o tym,
                    // którego kliknięcie uruchomi główne okno aplikacji
                    ShellToast toast = new ShellToast();
                    toast.Title = "SYNCHRONIZACJA";
                    toast.Content = "Konieczna jest synchronizacja danych";
                    toast.NavigationUri =
                        new Uri("/MainPhoneParentPage.xaml", UriKind.Relative);
                    toast.Show();

                    // Oznajmiamy do systemu,
                    // że obsługa zdarzenia została zakończona
                    NotifyComplete();
                }
                else
                    // Sprawdzenie i jeśli to konieczne pobranie
                    // informacji z serwera o przebiegu dnia dzieci
                    CheckDayEvents();
            }
        }
```

Ważnym faktem dotyczącym usług działających w tle pod kontrolą Windows Phone, jest to, że raz utworzone nie działają w nieskończoność, lecz gdy główne okno aplikacji nie jest włączane przez okres dwóch tygodni, to usługi te przestają być uruchamiane.

Dlatego też synchronizacja danych pomiędzy telefonem a serwerem wymaga włączenia głównego okna aplikacji. Tam też odbywa się sprawdzenie czy nie minął już termin ważności usługi.

W jaki sposób to się odbywa przedstawia poniższy fragment kodu:

```
    // Zwraca referencję do PeriodicTask jeśli taki istnieje
    // pośród zaplanowanych usług
    periodicTask = ScheduledActionService.Find(periodicTaskName) as PeriodicTask;

    // jeżeli ten agent już istnieje, to wykonujemy sprawdzenie
    // czy jego termin ważności nie zbliżył sie
    // jeśli tak to w celu uaktualnienia agenta musimy go usunąć
    // spośród zaplanowanych zdarzeń i utworzyć nowy obiekt
    if (periodicTask != null)
```

```
    {
        if ((periodicTask.ExpirationTime - DateTime.Now) < TimeSpan.FromDays(7))
            RemoveAgent(periodicTaskName);
        else
        {
            PeriodicStackPanel.DataContext = periodicTask;
            return;
        }
    }

    periodicTask = new PeriodicTask(periodicTaskName);
```

### 7.3.3 Patronus Parent

Aplikacja ta, zrealizowana jest jako „Windows Phone Pivot Application [21]", czyli składa się z kilku stron, pomiędzy którymi mamy możliwość nawigowania przesuwając ekran na boki.

Całością stron zarządza obiekt typu o nazwie jak można się domyśleć ***Pivot.*** Za pomocą tego obiektu mamy wpływ na to jakie strony są możliwe do wyświetlenia użytkownikowi, a także która wyświetlana jest obecnie – pozwala nawigować między nimi.

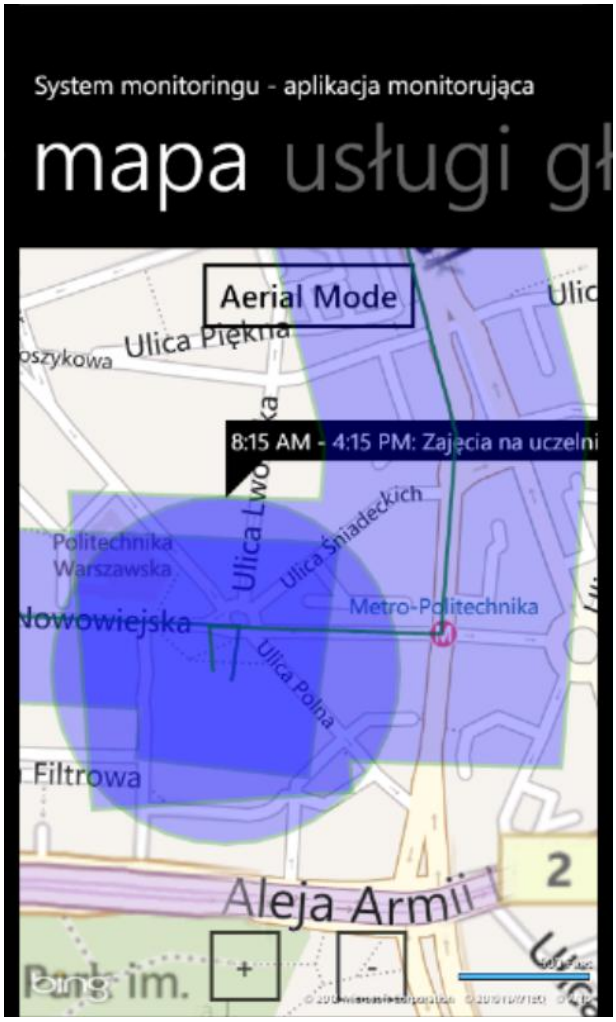
Jedynym wyjątkiem jest tutaj pierwsze uruchomienie aplikacji, kiedy to musimy podać dane użytkownika do pierwszego zalogowania się na serwerze. Wyświetlana jest wtedy jedynie jedna strona aplikacji.


**Rysunek 26 Pierwsze uruchomienie aplikacji rodzica, oraz informacja o błędzie logowania.**

Podanie danych użytkownika niezbędnych do logowania odbywa się wyłącznie raz. Przy każdym następnym uruchomieniu aplikacji, dane te pobierane są z lokalnej bazy danych.

Aplikacja rodzica składa się z następujących stron:

- Główna – informuje jako który użytkownik jesteśmy zalogowaniu w systemie, a także wyświetla informacje o ostatniej godzinie pobrania informacji z serwera, oraz podaje datę ostatniej synchronizacji. Podczas samej synchronizacji natomiast wyświetlane są poszczególne jej etapy.
- Dzieci - lista profili dzieci, pozwalająca przejść po wyborze profilu dziecka do jego planu dnia, lub do listy pozycji otrzymanych z serwera i mówiących o tym jak przebiega dzień dziecka.




Rysunek 27 Plan dnia dziecka na wybrany dzień pokazany na mapie (fragment)

- Plany dnia – uproszczone okno w porównaniu do tego jakie udostępnia aplikacja zarządzająca systemem, wyświetla dla wybranego dziecka zdefiniowane dla niego harmonogramy, oraz po wybraniu harmonogramu jego poszczególne wpisy. Mamy tutaj możliwość wyświetlenie wybranego harmonogramu na mapie, lub przejść do planu dnia jaki obowiązuje na dzień dzisiejszy.
- Plan na dziś – strona, która zawiera po przejściu do niej ze strony z planami dnia, listę wydarzeń zaplanowanych dla dziecka na dzisiejszy dzień. Możemy stąd przejść do

widoku mapy by zobaczyć wizualnie gdzie dziecko powinno się znajdować o jakiej porze.
- Przebieg dnia – tutaj znajdują się informacje, które przesłał telefon dziecka, dotyczące jak przebiega jego dzień. Komunikaty tutaj wyświetlone mówią o tym, czy był to zwyczajny odczyt danych, czy też może zgłoszenie alarmowe.
- Mapy – widok mapy zwiera informacje zależne od tego co chcemy tu wyświetlić. W porównaniu do kontrolki mapy używanej w aplikacji zarządzającej, ta tutaj jest o wiele uboższa (kontrolka ta jest wbudowana w jedną z bibliotek dostępnych tylko dla aplikacji Windows Phone).
- Usługi – strona diagnostyczna, informująca o pracy działających w tle serwisów, z których jeden odpowiada wyłącznie za pobieranie danych z serwera, na temat przebiegu dnia dzieci, a drugi dodatkowo sprawdza czy nie jest konieczne synchronizacja danych z serwerem.

Nad całością pracy aplikacji rodzica czuwa klasa ***Operations*** (wspominana wyżej), która stanowi warstwę logiki. Aplikacja rodzica natomiast reaguje na zdarzenia generowane przez klasę Operations, poprzez wyświetlenie nowych informacji, lub zasygnalizowanie końca synchronizacji danych.

#### 7.3.3.1 Obsługa mapy

Kolejnym aspektem aplikacji, któremu chciałbym się przyjrzeć to obsługa mapy [22]. Kontrolka mapy obsługiwana jest analogicznie do tej która znajduje się w aplikacji zarządzającej, aczkolwiek posiada ona mniejszą funkcjonalność. Również tutaj dodajemy informację na mapie działając na warstwach. Warstwy jakie tutaj wykorzystywane są dwie:

- PolygonLayer – warstwa na której wyświetlane są kształty obszarów (drogi, miejsca), wraz z opisami konkretnych obszarów. Źródłem danych dla tej warstwy jest lista obiektów typu ***Wpis_harmonogramu***.
- EventLayer – warstwa służąca do wyświetlania informacji o przebiegu dnia dla zadanego dziecka, źródłem dla niej jest lista obiektów typu ***Przebieg_dnia***.

Podczas tworzenia obiektów które będą następnie dodawane do wybranej warstwy na mapie, wywoływana jest metoda ***CheckMinMaxLocation***, której zadaniem jest znalezienie minimalnych i maksymalnych współrzędnych geograficznych wyświetlanego obszaru, aby później przy wyświetlaniu mapy było możliwe jest wyśrodkowanie właśnie na centrum obszaru.

### 7.3.4 Patronus Child

Interfejs do obsługi aplikacji dziecka jest dużo bardziej uboższy w porównaniu do aplikacji rodzica. Jest tak żeby zgodnie z założeniami, dziecko nie musiało podczas dnia wykonywać żadnych z nią interakcji.

Jedyną interakcją jaka będzie musiała być cyklicznie wykonywana w aplikacji dziecka jest synchronizacja danych z danymi na serwerze. Lecz taką synchronizację można przeprowadzać w domu i może jej asystować rodzic.

Sama aplikacja składa się z dwóch stron, i jej typ podobnie jak aplikacji rodzica opiera się na obiekcie ***Pivot***, który zarządza pracą ze stronami:

1. Główna – gdzie wyświetlane są informacje o tym jako jaki użytkownik jesteśmy zalogowani, o tym kiedy ostatni były przesyłane dane na serwer, ile pozycji nie jest wciąż wysłane, oraz kiedy była wykonywana ostatnia synchronizacja danych.
2. Plan dnia – strona składająca się z dwóch list:
    a. Planu dnia na dzisiejszy dzień.
    b. Lista nie wysłanych na serwer pozycji odczytanych z modułu GPS.


Rysunek 28 Główna strona aplikacji dziecka.

#### 7.3.4.1 Usługi działające w tle

Główna funkcjonalność aplikacji dziecka jest jednak wykonywana przez usługi działające w tle. Pominę tutaj wstęp do pracy z nimi, gdyż odbywa się to w ten sam sposób co na aplikacji monitorującej.

Różnica jest między nimi jednak taka, że w aplikacji monitorowanej zadanie cykliczne jest używane wg następującego scenariusza:

1. Sprawdzenie terminu ostatniego odczytu danych z modułu GPS i jeśli nie nadszedł termin sprawdzenia pozycji to kończenie wywołania usługi.
2. Sprawdzanie pozycji za pomocą obiektu GeoCoordinateWatcher, dodając obsługę zdarzeń wykonywanych po zmianie stanu modułu GPS i po odczytaniu nowej pozycji.
3. Zdarzenia po uruchomieniu obserwatora mogą być w uogólnieniu dwa: po pierwsze moduł GPS może zgłosić błąd, lub po drugie, uda się poprawnie odczytać pozycję.

4. W przypadku błędu działanie usługi jest kończone, nie odznaczając w systemie czasu odczytu pozycji (bo żadnego odczytu nie było).
5. W przypadku poprawnego odczytania pozycji, konieczne jest wykonanie kilku dodatkowych czynności
    a. Zatrzymanie modułu GPS, w celu oszczędzania energii, oraz zapewnienie jednego odczytu pozycji.

```
// zatrzymanie obserwatora pozycji
watcher.Stop();
// sprawdzanie w sekcji krytycznej czy trwa oczekiwanie na odczyt
// zdarzenia wywołane przez odczyt GPS mogą w teorii przyjść
// w bardzo bliskim odstępie czasu, a uwzględniając fakt, że
// uruchamiane są w osobnym wątku, to możemy spodziewać się dwóch odczytów
lock (mutex)
{
    // jeżeli okaże się, że w innym wątki był wykonany odczyt pozycji
    // to wychodzimy ze zdarzenia (bez NotifyComplete() bo tym się zajmie
    // pierwszy wątek)
    if (!checkingPosition)
        return;
    checkingPosition = false;
}
```

    b. Utworzenie obiektu przechowującego informacje o odczycie (***Przebieg_dnia***), zawierającego w tym momencie informacje o pozycji oraz czasie jej odczytu.
6. Za dalszą część pracy odpowiada już klasa ***Operations***, która sprawdza czy nadszedł już czas wysłania danych na serwer i jeśli tak to go wykonuje, a jeśli nie to jedynie zapisuje dane do lokalnej bazy danych, oraz wywołuje zdarzenie sygnalizujące koniec operacji.
7. Usługa działająca w tle czeka na odpowiedź z klasy ***Operations***, po czym kończy swoje działanie.

# 8 Testowanie

Testowaniu działania całego systemu chciałem poświęcić osobny rozdział, żeby podkreślić jak ważny jest ten etap, oraz opowiedzieć o tym ile potrafi wnieść do całego systemu. Głównie chciałbym się skupić na przypadkach testowych które pozwalają określić, czy dana funkcjonalność systemu może być uznana za działającą, czy nie. Następnie chciałbym przejść do testów aplikacji zarządzającej, oraz głównie aplikacji mobilnych.

Testowanie tych dwóch rodzajów aplikacji nie jest domyślnie wspierane przez Visual Studio i konieczne było doinstalowanie dodatków do niego, by możliwe było utworzenie projektu testowego dedykowanego na urządzenie z platformą Windows Phone [23].

## 8.1 Testowanie serwera i aplikacji Silverlight

Dwie rzeczy które są szczególnie ważne przy testowaniu aplikacji według mnie, są to po pierwsze własna wyobraźnia i pomysłowość w dobieraniu przypadków i danych testowych w celu zbadania jak największej możliwej liczby sytuacji mogących wystąpić podczas pracy aplikacji. Oraz po drugie – Debugger. Bez tego narzędzia testowanie byłoby mocno utrudnione, dlatego chciałem tutaj podkreślić jak jest on przydatny, gdyż czasem nawet najbardziej przemyślane wizje programisty, wymagają prześledzenia wykonania kodu kod po kroku by zrozumieć i odpowiednio zanalizować wykonany kod w celu znalezienie błędu.

Podczas stopniowego dodawania coraz to nowych funkcjonalności do aplikacji internetowej, oraz często wiążących się z tym modyfikacji usług sieciowych, udało się znaleźć przypadki testowe, które pokazują obecnie już, że aplikacja działa poprawnie, a wcześniej dawały możliwość znalezienie błędów.

Zgodnie z ogólnie przyjętymi zasadami testowania, rozpoczyna się ono od testowania pojedynczych funkcjonalności (testy jednostkowe), następnie przechodząc w etap testowania całych modułów, oraz testy integracyjne całego systemu.

W praktyce dało się zaobserwować, że czasem zdarzało się tak, że poprawa jednego błędu, powodowała inny, zatem, po każdej poprawce niezbędne było wykonanie już wcześniej poprawnie wykonywanych przypadków testowych.

Sprawdzanie poprawności funkcji udostępnianych przez aplikację zarządzającą pokazało, że konieczne było dodanie m.in. takich funkcji i zabezpieczeń podczas pracy aplikacji:

1. Wczytanie wszystkich danych, które mają być wyświetlane w aplikacji do pamięci podręcznej podczas startu aplikacji, co pozwala uniknąć sytuacji gdy znajdziemy przypadek testowy, w którym np. okaże się że pewien ***ComboBox*** jest niewypełniony danymi. Wcześniej zakładałem koncepcję, że dane z serwera będą pobierane wtedy kiedy będą one konieczne do wyświetlenia, w zależności na którą stronę aplikacji przejdzie użytkownik.

   Niestety okazało się, że nawet proste założenie, żeby za każdym razem przed wyświetleniem danych sprawdzać, czy są one pobrane i jeśli nie to je pobrać komplikuje kod. Dlaczego? Ponieważ wywołania usług sieciowych odbywają się asynchronicznie, zatem dla każdego takie sprawdzenia konieczne by było utworzenie odpowiedniego zdarzenie w kontrolerze, który by pobrał te dane, a następnie obsłużenie go po stronie widoku, oraz wyświetlenie formularza oczekiwania w czasie gdy serwer będzie przygotowywał te dane.

Prostszym i bardziej przejrzystym do czytania w kodzie okazało się więc rozwiązanie z wczytywaniem danych po zalogowaniu do aplikacji. Zalety tego są takie, że jak już wspomniałem, nie musimy się martwić przy wyświetlaniu, że jakichś danych nie mamy pobranych, po drugie użytkownik podczas pracy z aplikacją nie będzie musiał oczekiwać na odpowiedź serwera gdy już się zaloguje i będzie chciał wyświetlić stronę z innymi danymi.

2. Zabezpieczenia przed usuwaniem danych, które są używane gdzie indziej. Mianowicie, nie powinno być możliwe usuwanie regionów, które służą do tworzenia obszarów, czy też obszarów, które są używane we wpisach harmonogramów. W takich sytuacjach użytkownik otrzymuje komunikat co się stało i dlaczego usunięcie danych nie jest możliwe.
3. Zablokowanie dodawania monitorowanych profili dzieci użytkownikom, którzy nie mają uprawnień rodzica, a także odwrotna sytuacja, monitorowania użytkowników którzy funkcjonują w systemie jako dzieci.
4. Usuwanie harmonogramu przypisanego do danego użytkownika (dziecka), nie powoduje od razu usunięcia go, gdyż możliwe jest, że jest on używany przez innego użytkownika. Samo jednak usunięcie harmonogramu przypisanego do użytkownika jest sytuacją poprawną i nie powinno powodować wyświetlenia komunikatu, zatem operacja ta zawsze się wykonuje (usunięcie powiązania między użytkownikiem, a harmonogramem), oraz dodatkowo odbywa się sprawdzenie czy harmonogram nie jest używany przez kogoś innego. Dopiero po stwierdzeniu, że nie jest on używany przez nikogo, zostaje on usunięty z bazy danych, dzięki czemu nie będzie tam panował przysłowiowy bałagan.

### 8.1.1 Wyświetlanie map

Testowanie tej funkcjonalności w zasadzie sprowadza się do dwóch rzeczy:

1. Sprawdzenie poprawności wyliczanych współrzędnych punktów wyświetlanych na mapie.
2. Subiektywnej oceny czy prezentacja punktów jest wyraźna i przejrzysta.

O ile pierwszy punkt daje się zautomatyzować w postaci testów jednostkowych dla poszczególnych metod, o tyle punkt drugi każdy użytkownik będzie mógł ocenić inaczej.

Sposób w jaki mógłby ten punkt być przetestowany to wystawienie wersji beta systemu dla pewnego zróżnicowanego grona użytkowników i zebranie ich opinii.

Testowanie natomiast pierwszego punktu będzie opisane w kolejnym podrozdziale, gdyż te same funkcjonalności wyznaczania obszarów na mapie wykonywane są w aplikacji mobilnej rodzica. Ponadto sposób wykonywania automatycznych testów na emulatorze może stanowić pewnego rodzaju ciekawostkę.

### 8.1.2 Automatyzacja testów

Projekt ***MonitoringSystemTest*** zawiera testy jednostkowe sprawdzające poprawność działania ważniejszych metod wystawianych przez usługę sieciową.

Uruchomienie ich odbywa się z poziomu Visual Studio z menu *Test/Run/All Tests in Solution*. Testy wymagają aby był włączony serwer bazy danych, ponieważ z niej korzystają.

Do bazy danych wprowadzane są dane testowe, na który operują testy. Tworzona jest nowa grupa użytkowników systemów, tak aby nie było możliwości ingerencji w rzeczywiste dane w bazie.

Po zakończeniu testowania, wprowadzone dane testowe są usuwane z bazy.

Nie opisuje tutaj przypadków testowych zawartych w projekcie, gdyż informacje o tym znajdują się w dokumentacji załączonej do pracy jak i w samym kodzie w postaci komentarzy.

## 8.2 Testy aplikacji mobilnej

Większość logiki zarówno aplikacji monitorowanej jak i monitorującej znajduje się w bibliotece ***PhoneOperations***. Ona to odpowiada za przygotowanie danych do prezentacji, a także za wyznaczenie współrzędnych dla granic obszarów wyświetlanych na mapie, oraz za sprawdzanie czy pozycja o zadanych współrzędnych mieści się w granicach obszaru, w którym dziecko powinno przebywać o zadanej godzinie.

### 8.2.1 Użycie emulatora

Niezbędne podczas tworzenia aplikacji mobilnych okazało się użycie emulatora urządzenia z systemem Windows Phone, który pozwala na swobodne testowanie większości funkcjonalności prawdziwego telefonu.

Emulator pozwalał sprawdzić poprawność wyświetlanych danych, zarówno po ich synchronizacji, czy też pobraniu z serwera, a także podczas nawigacji pomiędzy różnymi stronami aplikacji.

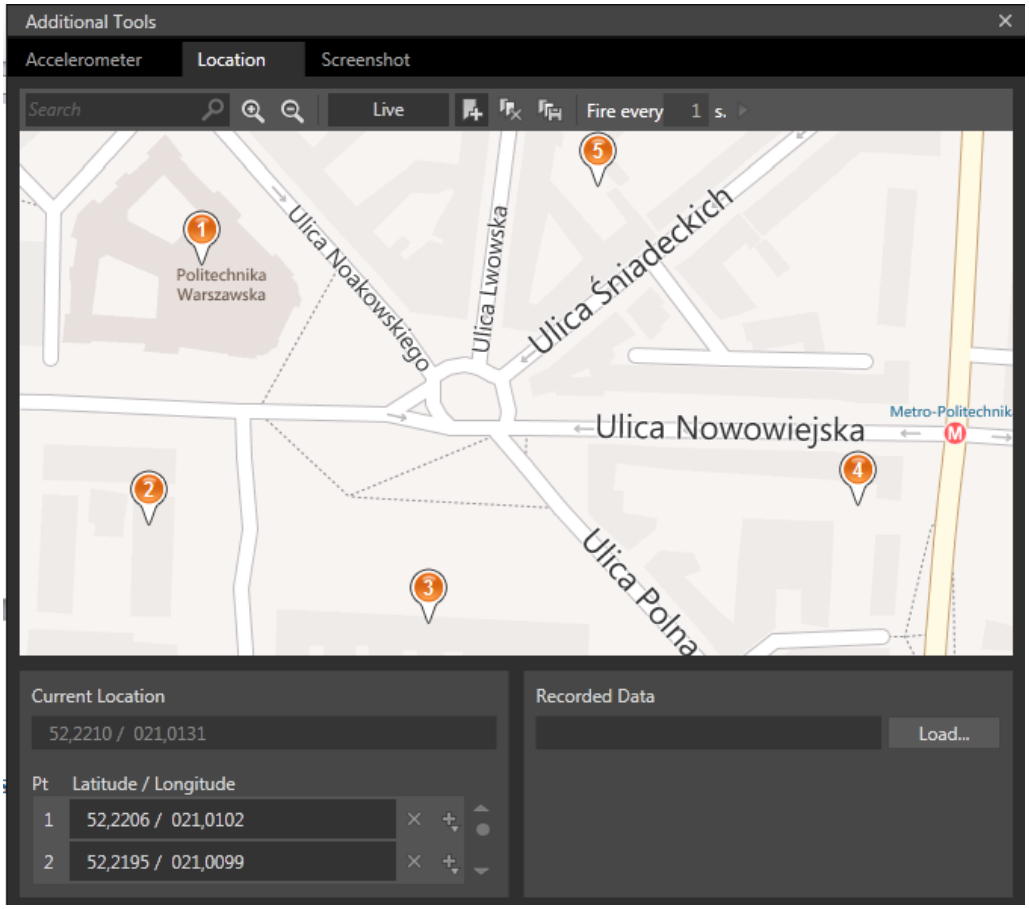

Rysunek 29 Emulatora urządzenia z systemem Windows Phone

Warto wspomnieć o kilku scenariuszach jakie były pomocne podczas testowania:

1. Próba logowania, lub uruchomienia synchronizacji przy nieaktywnym połączeniu z serwerem (wyłączony serwer projektowy ASP.NET), oraz analogiczna sytuacja przy połączeniach uruchamianych przez serwisy działające w tle.
2. Sprawdzenie krokowego wykonania zadań w tle używając Debugger'a, oraz pomocnej funkcjonalności tych serwisów, która na potrzeby testów pozwala uruchamiać je co określony interwał czasu.

```
// Jeśli włączony jest tryb debuggowania, to uruchom zdarzenie w przeciągu minuty.
#if DEBUG_AGENT
  ScheduledActionService.LaunchForTest(periodicTaskName,
TimeSpan.FromSeconds(60));
#endif
```

3. Poprawne zakończenie metod w przypadku błędów (np. usługa działająca w tle zarówno przy poprawnym jak i niepoprawnym zakończeniu powinna wywołać ***NotifyComplete***()), oraz ponowna próba wykonania metody przy kolejnym wywołaniu serwisu.
4. Także zapis o terminie zakończenia synchronizacji, czy też godziny pobrania informacji o pozycji dzieci z serwera powinien być wykonywany na koniec wykonywania całej operacji, gdy zakończy się ona poprawnie, a w przeciwnym wypadku taki zapis nie powinien się dokonać, ale operacja powinna się zakończyć, nie przerywając pracy całej aplikacji.




**Rysunek 30 Dodatkowa funkcjonalność emulatora - wysyłanie pozycji**

5. Sprawdzenie czy używanie przycisków nawigacyjnych takich jak po wybraniu dziecka z listy ***zdarzenia***, czy ***plan dnia***, ma nas przenieść na odpowiednią stronę i uzupełnić ją

danymi dotyczącymi wybranego dziecka. Dotyczy to też przenoszenia nas na stronę z widokiem mapy.

6. Szczególnym przypadkiem jest sprawdzenie odczytu pozycji z modułu GPS przez agenta działającego w tle. Problemami tutaj jest fakt, że nie używając trybu Debug nie widzimy kiedy rozpoczyna się wywołanie usługi, oraz drugi problem to zasymulowanie odczytu pozycji z modułu GPS.

Tutaj jedna z pomocą przychodzi nam dodatkowa funkcjonalność emulatora, która pozwala właśnie takie zdarzenia wysyłane do modułu GPS wysyłać. Mamy z poziomu tego dodatku dostęp także do akcelerometru i możliwość tworzenia zrzutów ekranu z ekranu emulatora.

Wracając jednak do symulacji przemieszczania się, działa to w sposób bardzo prosty. W widoku mapy musimy zaznaczyć punkty w przestrzeni, których współrzędne geograficzne zostaną przesłane do modułu GPS emulatora. Dodatkowo możemy raz utworzone punkty zapisać w pliku XML w celu ich późniejszego użycia ponownie. Po ustawieniu wszystkich punktów, mamy możliwość określenia co ile sekund nastąpi odpalenie kolejnego punktu, tak by wywołał na emulatorze zdarzenie związane ze zmianą pozycji.

Przy pierwszych próbach napotkałem na pewnie trudności z synchronizowaniem czasu uruchomienia obserwatora pozycji na telefonie, z uruchomieniem wysyłania pozycji do modułu GPS.

### 8.2.2 Automatyzacja testów

Będąc przy temacie odczytywania pozycji geograficznej urządzenia chciałbym przejść do sposobu sprawdzania jej zgodności z harmonogramem, oraz testowania pozostałych funkcjonalności związanych z wyznaczaniem współrzędnych m.in. granic obszaru wokół zdefiniowanego w harmonogramie obszaru.

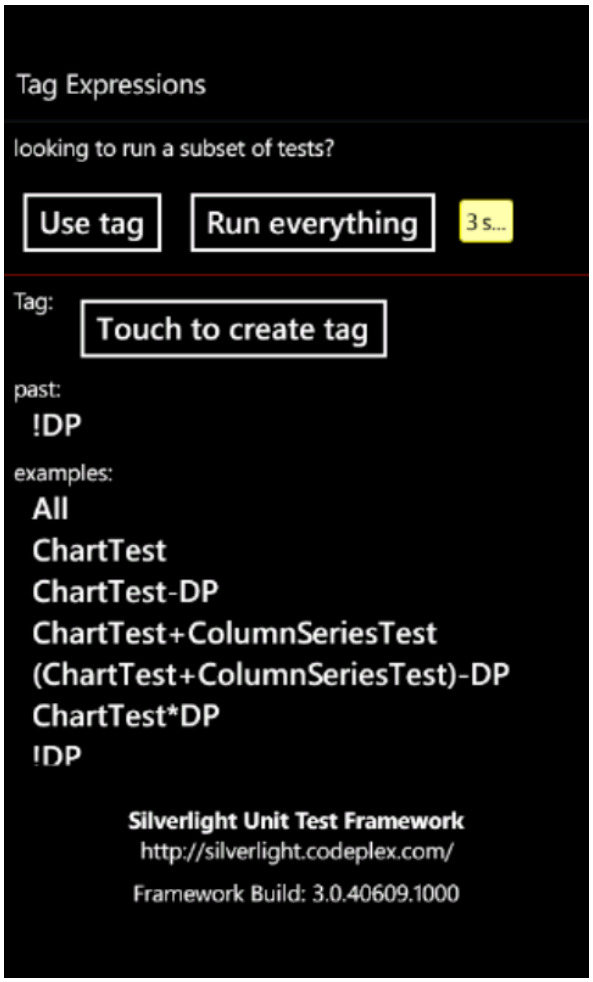
Możliwe jest utworzenie projektu testowego dla platformy Windows Phone, jednak aby tego dokonać, konieczne jest najpierw zainstalowanie dodatku do Visual Studio o nazwie NuGet [24], który to pozwala na swobodne zarządzanie dodatkami, rozszerzeniami i bibliotekami zainstalowanymi w Visual Studio.

Przechodząc do części właściwej konieczne jest używając NuGet zainstalowanie paczki rozszerzeń o nazwie WindowsPhoneEssentials.Testing, która do daje do istniejących szablonów projektów projekt testowy dla Windows Phone (także w wersji 7.5 Mango).

Organizacja testów w Visual Studio polega na oznaczeniu testowych klas atrybutem ***[TestClass]*** , oraz metod testowych atrybutem ***[TestMethod]*** . Przygotowany szablon testowy zawiera też aplikację zarządzającą testami i ich wykonywaniem, która może być wdrożona zarówno na emulator jak i rzeczywiste urządzenie. Możliwe są też do użycia inne atrybuty dotyczące metod, które pozwalają oznaczyć je jako uruchamiane przed wykonaniem wszystkich testów w klasie, oraz po wykonaniu wszystkich, lub też nakazujące uruchamianie metody przed, lub po każdej metodzie testowej. Dzięki czemu za pomocą tych metod inicjalizacyjnych (***[TestInitialize], [ClassInitialize]***), oraz sprzątających (***[TesCleanup], [ClassCleanup]***), możliwe jest przygotowanie danych testowych, oraz usunięcie ich by nie ingerowały w normalny tryb pracy urządzenia, czy też po prostu przygotować obiekty do użycia.

Sposób w jaki zostały zorganizowane testy bibliotek działających na urządzeniu mobilnym, wygląda podobnie jak to ma miejsce w testach aplikacji serwerowej, mianowicie:

1. Przed rozpoczęciem testów, zawartość lokalnej bazy danych jest wczytywana do pamięci podręcznej
2. Baza danych jest czyszczona i wprowadzane są do niej dane testowe.
3. Następuje właściwe wykonanie testów.
4. Dane testowe są usuwane i ponownie do bazy wprowadzane są dane sprzed uruchomienia testów.



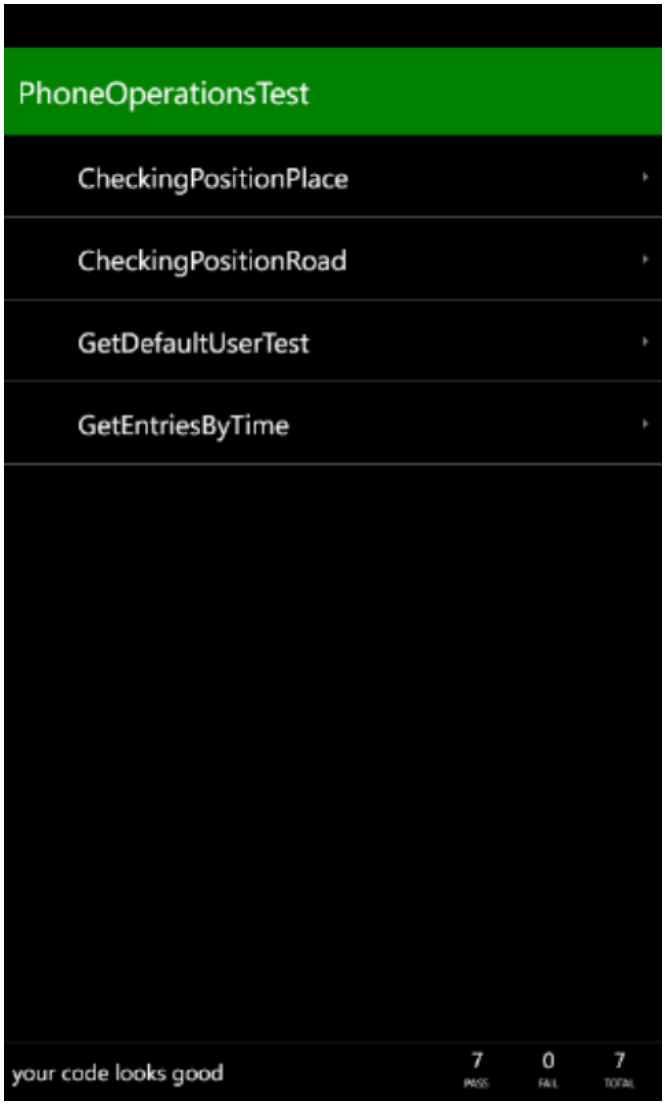

**Rysunek 31 Okno główne aplikacji testującej**

Funkcjonalności, które są testowane w tym projekcie dotyczą typowo operacji opierających się na wyznaczaniu pozycji (czyli dotyczą klasy ***GeoCodeCalc***), lub też sprawdzają poprawność działania klasy ***Operations***. Dlatego też dla każdego z tych przypadków została utworzona klasa testowa.

Sprowadza się to do tego, że testy wyznaczania pozycji dotyczą następujących funkcjonalności:

1. Wyznaczania dystans pomiędzy dwoma współrzędnymi geograficznymi.
2. Wyznaczania współrzędnych punktów leżących na okręgu o zadanym środku i promieniu podanym w metrach, oraz sprawdzaniu czy zadana współrzędna geograficzna należy do okręgu.
3. Wyznaczanie współrzędnych wierzchołków figury, która tworzy obszar otaczający łamaną reprezentującą drogę, oraz sprawdza czy zadane punkty geograficzne należą do tego obszaru.

Druga klasa testowa skupia się na warstwie operacji:




Rysunek 32 Raport z testów dla wybranej klasy testującej

1. Test komunikacji z bazą danych, wczytujący domyślnego użytkownika dla aplikacji.
2. Wyznaczanie wpisów harmonogramu obowiązujących dla zadanej godziny.
3. Sprawdzanie dla zadanej pozycji geograficznej i godziny czy zgadza się ona z harmonogramem (dwa przypadki, dla drogi i dla miejsca).

### 8.2.3 Testy na rzeczywistym urządzeniu

Pomimo sprawdzenia działania pojedynczych funkcjonalności podczas wcześniejszych testów (testy jednostkowe), oraz symulowania pracy rzeczywistego telefonu poprzez emulator i przesyłanie mu danych ze współrzędnymi geograficznymi do modułu GPS, a także sprawdzenie podczas osobnych testów działania aplikacji rodzica na emulatorze i wprowadzanie w tym czasie danych testowych, które reprezentują odczyt pozycji przez aplikację dziecka, to nie możliwe jest na jednym komputerze uruchomienie dwóch, lub więcej instancji emulatora. A właśnie taka sytuacja zwykle będzie miała miejsce podczas zakładanego wyglądu pracy systemu.

Niestety urządzenia z systemem Windows Phone Mango, są obecnie dość drogie, i nie był możliwy zakup dwóch takich urządzeń do celów testowych. Mimo wszystko jednak moja praca

inżynierska skupiała się na prawidłowym zaprojektowaniu kompletnego systemu monitoringu, zaś zaimplementowana aplikacja stanowi jedynie jego wycinek, aczkolwiek zawierający główne funkcjonalności (czego dowodem mogą być przedstawione wcześniej testy).

Oczywiście żadne testy nie oddadzą tego jak cały system zadziała w rzeczywistym środowisku, w którym mogą być napotkane problemy i sytuacje, których nie możliwe jest zbadanie jedynie używając emulatora.

Zatem temat testów na rzeczywistym urządzeniu jest wciąż otwarty i mam nadzieję, że z czasem gdy będę miał do takowych dostęp, już co prawda dla własnej satysfakcji, lecz chciałbym sprawdzić zachowanie systemu w praktyce.

Będzie do tego jednak mi potrzebny serwer udostępniający usługi sieciowe w Internecie. Uzyskać to można korzystając z usługi Dynamicznego DNS [25], o której wspominałem już wcześniej. Udostępniony w taki sposób serwis w zupełności wystarczy przynajmniej na potrzeby testów na rzeczywistym urządzeniu.

# 9 Podsumowanie

Projekt o nazwie Patronus zakończył się sukcesem. System działa i został przetestowany na środowisku testowym z użyciem emulatora urządzenia z Windows Phone. Zaimplementowane zostały funkcjonalności zgodnie z założeniami jakie miała spełniać pierwsza wersja systemu.

Jednak warto zwrócić uwagę, że proces projektowania i wykonania systemu informatycznego jest bardzo złożony, oraz w przypadku projektów rzeczywistych, a nie tylko akademickich, nie da się go zamknąć w określonych ramach, gdyż zawsze wytwarzanie oprogramowania zahacza o różne dzieciny wiedzy, nie tylko typowo programistyczne.

Nie inaczej było z systemem monitoringu Patronus. Konieczne było zapoznanie się z tym co obecnie funkcjonuje na rynku, zastanowienie się jak taką usługę wyobrażałby sobie przeciętny rodzic w Polsce, ale też rozważyć funkcjonalności jaki mogliby wymyśleć bardziej wymagający rodzice.

Sam projekt zaproponowany przeze mnie może być jeszcze dużo bardziej rozwinięty, gdyż gdyby doszło do jego rzeczywistego wdrożenia na rynek to sami rodzice mogliby proponować co według nich byłoby przydatnie jeszcze w systemie.

Wśród rozmaitych rozszerzeń mogłaby się znaleźć obsługa przynależności użytkowników do wielu grup, rozbudowa sposobu określania obszarów na mapie (tworzenie obszarów złożonych, nie tylko koło i łamana), czy rozszerzenie sposobu tworzenia harmonogramów, możliwość ich elastyczniejszego wielokrotnego użycia. Pomysły na rozszerzenie systemu jest wiele i pewnie mogłoby być ich jeszcze więcej.

Bardzo ciekawą częścią mojej pracy było tworzenie aplikacji mobilnej, które wygląda nieco inaczej niż tworzenie aplikacji desktopowych czy internetowych. Nie chciałbym się przy zakończeniu rozwodzić na tym co już została zrobione w systemie, lecz wolałbym zaznaczyć, że aplikacje te wciąż kryją w sobie niewykorzystany potencjał, na co pozwala interesujący według mnie system Windows Phone i mnogość funkcji i zastosowań smartfonów, które może nam podyktować wyobraźnia. Chociażby użycie wymienionego w projekcie akcelerometru, lub zastosowanie większych możliwości personalizacji aplikacji, zarówno pod względem wyglądu, wyboru sposobu alarmowania użytkownika, czy też wprowadzanie aplikacji w tryb nocny, gdzie nie powinny wykonywać się żadne akcje, w celu mniejszego zużycia baterii.

Kończąc podsumowanie mojej pracy, mam nadzieje, że zainteresowała ona czytelnika tematyką zarówno tworzenia oprogramowania, jak też możliwościami aplikacji mobilnych.

# 10 Bibliografia


[1] „Navitracker,” [Online]. Available: http://www.navitracker.pl/index.php.

[2] „Navitacer,” [Online]. Available: http://www.navitracer.com.

[3] „Gdzie Jest Dziecko - usługa lokalizacyjna,” [Online]. Available: https://gdziejestdziecko.pl/public/sc01-index.jsf.

[4] „SpySat,” [Online]. Available: http://spysat.pl/.

[5] „YourCargo,” [Online]. Available: http://www.yourcargo.org/frontend/funkcjonalnosci.html.

[6] [Online]. Available: http://www.t-mobile.pl/pl/biznes/stali_klienci/uslugi_dla_firm/monitoring_sim.

[7] [Online]. Available: http://nawigacja.orange.pl/nawigator.html.

[8] K. Sacha, Inżynieria oprogramowania, Warszawa: Wydawnictwo Naukowe PWN, 2010.

[9] „Silverlight Bing Maps: Draw Circle Around a Latitude/Longitude Location,” [Online]. Available: http://pietschsoft.com/post/2010/06/28/Silverlight-Bing-Maps-Draw-Circle-Around-Latitude-Longitude-Location.aspx.

[10] „.NET Framework,” [Online]. Available: http://pl.wikipedia.org/wiki/.NET_Framework.

[11] „Silverlight,” [Online]. Available: http://pl.wikipedia.org/wiki/Microsoft_Silverlight.

[12] „CollabNet,” [Online]. Available: http://ankhsvn.open.collab.net/.

[13] „Entity Framework,” [Online]. Available: http://msdn.microsoft.com/en-us/data/aa937723.

[14] „Windows Communication Foundation,” [Online]. Available: http://en.wikipedia.org/wiki/Windows_Communication_Foundation.

[15] „Entity Framework - aplikacja trójwarstwowa,” [Online]. Available: http://msdn.microsoft.com/pl-pl/library/ff714342.aspx.

[16] „Making a Service Available Across Domain Boundaries,” [Online]. Available: http://msdn.microsoft.com/en-us/library/cc197955(v=vs.95).aspx.

[17] „Enabling SSL for a WCF Service,” Christophe Geers' Blog, [Online]. Available: http://cgeers.com/2012/01/30/enabling-ssl-for-a-wcf-service/.

[18] „Bing Maps,” [Online]. Available: http://msdn.microsoft.com/en-us/library/dd877180.aspx.

[19] „SQL Server Compact Toolbox,” [Online]. Available: http://sqlcetoolbox.codeplex.com/.

[20] „Background Agents Overview for Windows Phone,” [Online]. Available: http://msdn.microsoft.com/en-us/library/hh202942(v=vs.92).aspx.

[21] „Pivot Control for Windows Phone,” [Online]. Available: http://msdn.microsoft.com/en-us/library/ff941123(v=vs.92).

[22] „Bing Maps Silverlight Control for Windows Phone,” [Online]. Available: http://msdn.microsoft.com/en-us/library/ff941096(v=vs.92).

[23] „Windows Phone Test Project,” [Online]. Available: http://visualstudiogallery.msdn.microsoft.com/6819514d-4bd6-4f31-a231-48c6530ed03b.

[24] „NuGet,” [Online]. Available: http://nuget.codeplex.com/wikipage?title=Getting%20Started.

[25] „DynDNS,” [Online]. Available: http://www.dipol.com.pl/do_czego_sluzy_ddns_dynamic_domain_name_system_i_jak_go_uzywac__bib93.htm.

[26] „Pelikan,” [Online]. Available: http://www.orange.pl/kid,4002009272,id,4002009274,title,Pelikan,article.html.

[27] „Oferta abonamentu,” [Online]. Available: http://www.t-mobile.pl/pl/indywidualni/taryfy/abonament/bez_telefonu/wyjatkowa_okazja.

[28] „JSON,” [Online]. Available: http://pl.wikipedia.org/wiki/JSON.

[29] „XML,” [Online]. Available: http://pl.wikipedia.org/wiki/Xml.

[30] „Visio 2010,” [Online]. Available: http://office.microsoft.com/pl-pl/visio/.

[31] „Determine if the point is in the polygon,” [Online]. Available: http://social.msdn.microsoft.com/forums/en-US/winforms/thread/95055cdc-60f8-4c22-8270-ab5f9870270a/.

[32] „Silverlight Class Library Project Template,” [Online]. Available: http://visualstudiogallery.msdn.microsoft.com/bd30c036-01f2-42ff-9b32-25010a6226f7.

# 11 Spis załączników

- Płyta CD zawierająca następujące katalogi:
  - Patronus.NET – kod źródłowy systemu
  - Patronus Documentation - dokumentacja kodu źródłowego
  - Instalation – instrukcja instalacji systemu i wdrożenia aplikacji na urządzenia mobilne
  - User manual – instrukcja obsługi systemu Patronus

# 12 Spis ilustracji

Rysunek 1. Strona główna Navitracer ..... 16
Rysunek 2. Strona główna usługi lokalizacyjnej Gdzie Jest Dziecko ..... 17
Rysunek 3. Opis protokołu używanego do komunikacji z serwerami SpySat ..... 18
Rysunek 4. Schemat działania systemu SpySat ..... 19
Rysunek 5. Monitoring Plus - schemat działania ..... 20
Rysunek 6. Konfigurowanie i zarządzanie systemem ..... 25
Rysunek 7. Konfiguracja urządzeń mobilnych ..... 26
Rysunek 8. Synchronizacja danych - aplikacje mobilne ..... 27
Rysunek 9 Synchronizacja danych – użytkownicy ..... 27
Rysunek 10. Monitorowanie położenia dziecka z telefonu rodzica. ..... 28
Rysunek 11. Monitorowanie na telefonie dziecka ..... 29
Rysunek 12. Schemat systemu ..... 33
Rysunek 13 Schemat komponentów serwera ..... 34
Rysunek 14 Schemat bazy danych serwera ..... 35
Rysunek 15 Schemat wywołań metod z usług sieciowych serwera ..... 40
Rysunek 16 Obszary wokół dwóch punktów drogi ..... 49
Rysunek 17 Obszar wokół drogi składającej się z trzech punktów ABC ..... 50
Rysunek 18 Schemat aplikacji mobilnej dziecka ..... 51
Rysunek 19 Schemat aplikacji monitorującej ..... 55
Rysunek 20 Podział systemu na projekty ..... 60
Rysunek 21 Model bazy danych utworzony za pomocą Entity Framework ..... 62
Rysunek 22 clientaccesspolicy.xml ..... 63
Rysunek 23 crossdomain.xml ..... 63
Rysunek 24 Zarządzanie użytkownikami i harmonogramami. ..... 64
Rysunek 25 Przykładowy region w widoku mapy (droga wraz z obszarem dookoła) ..... 67
Rysunek 26 Pierwsze uruchomienie aplikacji rodzica, oraz informacja o błędzie logowania. ..... 72
Rysunek 27 Plan dnia dziecka na wybrany dzień pokazany na mapie (fragment) ..... 73
Rysunek 28 Główna strona aplikacji dziecka. ..... 75
Rysunek 29 Emulatora urządzenia z systemem Windows Phone ..... 79
Rysunek 30 Dodatkowa funkcjonalność emulatora - wysyłanie pozycji ..... 80
Rysunek 31 Okno główne aplikacji testującej ..... 82
Rysunek 32 Raport z testów dla wybranej klasy testującej ..... 83