PIOTR JASTRZĘBSKI • PIOTR BARTOSZUK • PIOTR OKUŁA • WOJCIECH KACZOROWSKI

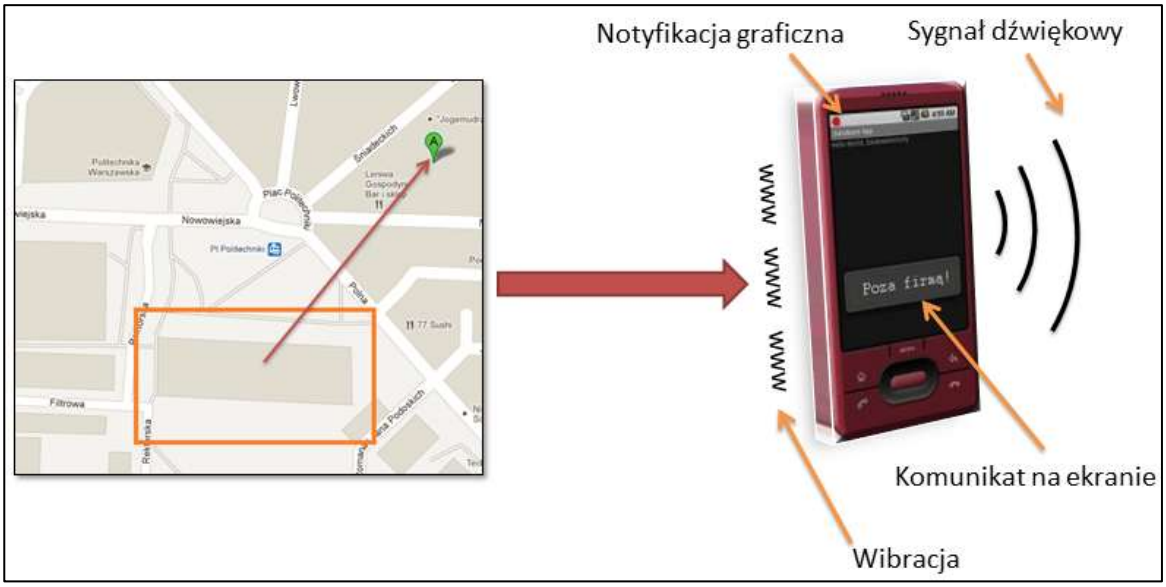
PROJEKT TIN 2011• OPIEKUN: DR INŻ.PIOTR GAWKOWSKI

SPIS TREŚCI

OPIS FUNKCJONALNOŚCI

Dla końcowego użytkownika widoczne będą 2 z 3 przygotowanych modułów.

1. Aplikacja Android – program działający w tle w formie *Service* nie wyświetlający żadnych informacji do momentu opuszczenia przez użytkownika wyznaczonej strefy. Na podstawie przesyłanych komunikatów o współrzędnych położenia i otrzymanych od serwera ograniczeniach wynikających z zasięgu strefy aplikacja stwierdza czy użytkownik znajduje się w dozwolonym obszarze, czy też nie. Jeśli nie – telefon poinformuje go o tym wibracją, sygnałem dźwiękowym, monitem w formie *Android Toast* oraz notyfikacją graficzną na głównej belce systemu.



W momencie opuszczenia dozwolonej strefy, użytkownik ostrzegany jest o tym fakcie przez aparat telefoniczny. W podglądzie aplikacji zdalnej operator może śledzić jego położenie na mapie.

2. Aplikacja webowa – zapewnia możliwość przedstawienia graficznego na mapie pozycji użytkowników telefonów. Ułatwia wyznaczanie obszarów dozwolonych (tu – pomarańczowe wielokąty). Zapewnia możliwość wysłania komunikatów przywołania.

Przykładowy interface aplikacji webowej. Zapewnia możliwość wyznaczania obszarów i śledzenia użytkowników.

WYMAGANIAFUNKCJONALNE

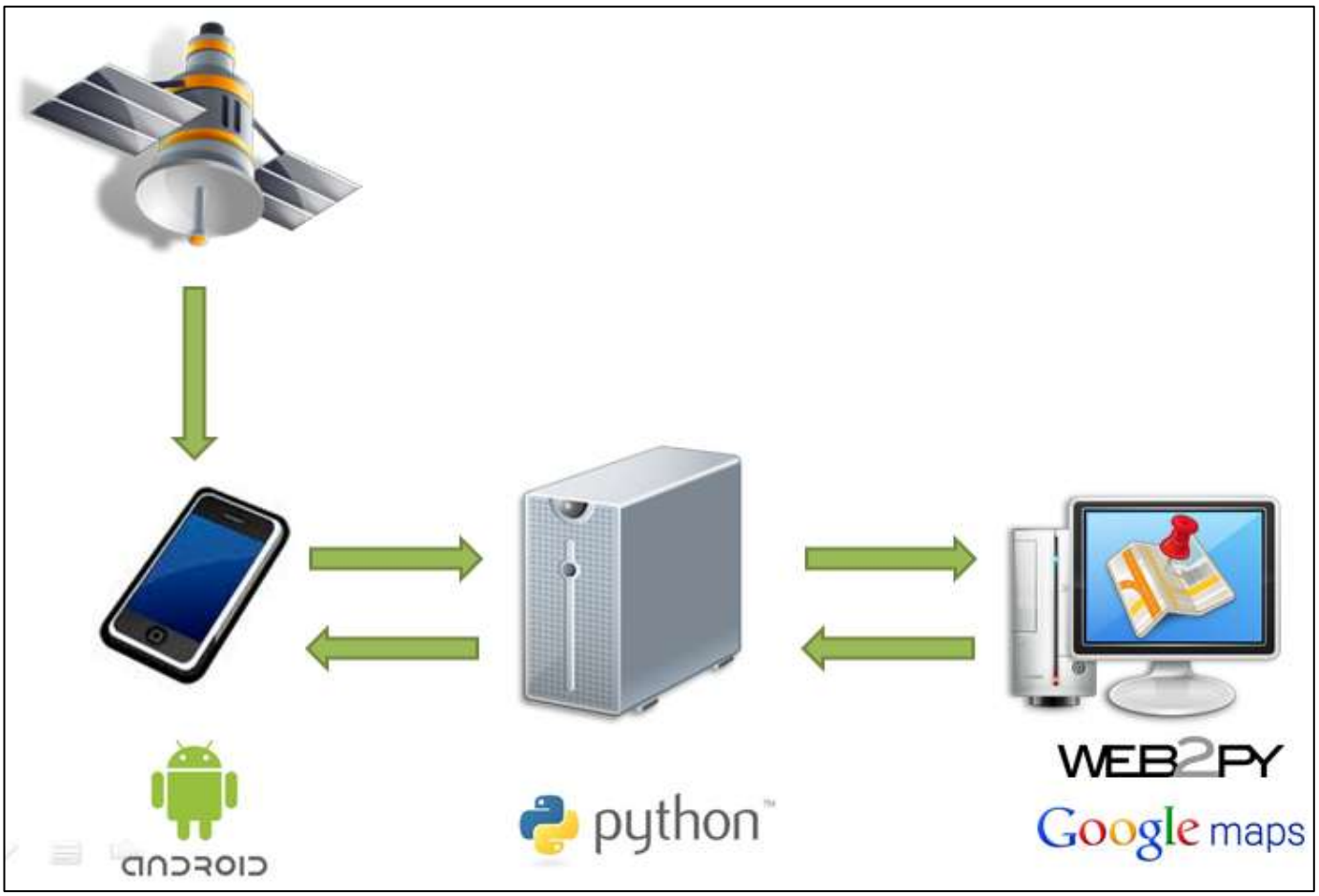
ID	Funkcja	Opis	Priorytet
1	Funkcje dotyczące telefonu		
1.1	Zbieranie współrzędnych	Zbieranie koordynatów zarówno z GPSu, jak i ze wskazań widoczności nadajników sieci Wi-Fi w celu ustalenia maksymalnie dokładnej pozycji	Wysoki
1.2	Analiza położenia	W zależności od położenia zmiana częstotliwości wysyłania danych do serwera. Analizowanie czy telefon jest nadal w strefie dozwolonej	Wysoki
1.3	Informowanie o opuszczeniu strefy	W momencie opuszczenia strefy włączenie wszystkich mechanizmów informowania użytkownika o konieczności powrotu.	Średni
2	Funkcje dotyczące serwera		
2.1	Komunikacja	Nawiązania i obsługa komunikacji dwustronnej Serwer – Telefon oraz Serwer – Aplikacja webowa	Wysoki
3	Funkcje dotyczące aplikacji webowej		
3.1	Aktualizacja na mapie	Funkcja odwzorowująca pozycję śledzonych użytkowników na mapie	Wysoki
3.2	Wyznaczenie obszaru	Wyznaczenie wielokąta stanowiącego obszar, w którym będzie mógł przebywać użytkownik bez wywołania alarmu opuszczenia strefy	Wysoki
3.3	Przywołanie użytkownika	Wprowadzenie i wysłanie do zadanego użytkownika komunikatu	Niski
3.4	Synchronizacja danych	Przesłanie danych i informacji wprowadzonych po stronie aplikacji webowej w stronę serwera pośredniczącego	Wysoki

WYMAGANIA NIEFUNKCJONALNE

- Racjonalny czas odpowiedzi aplikacji (zarówno po stronie telefonu jak i aplikacji webowej), umożliwiający pracę interaktywną.
- Przez zastosowanie wielowątkowego serwera możliwość komunikacji z większą ilością użytkowników i równoległe przetwarzanie otrzymywanych informacji.
- Możliwość niezawodnego gromadzenia danych niezależnie lub zależnie w mniejszym stopniu od widoczności satelitów na niebie (pozycjonowanie na podstawie analizy sygnału sieci WLAN)
- Graficzny interfejs użytkownika musi podnieść walory użytkowe programu i ułatwić wdrożenie. Prezentacja danych i codzienna obsługa musi być intuicyjna i prosta, a z każdą opcją powiązana powinna być czytelna ikona lub prosty, jednoznaczny opis.
- Niskie wymagania co do pamięci RAM – minimalny interfejs typu *Android Service*
- Zmienne co do czasu interwały przesyłania informacji w sieci do serwera, uzależnione od przebywania w środku lub na zewnątrz strefy dozwolonej
- Ze względu na koszta, zredukowana do minimum ilość informacji przesyłanych przez sieć

ARCHITEKTURA

Projekt Android Spy Application jest złożony z kilku elementów: aplikacji klienckiej uruchamianej na telefonie z systemem Android oraz części serwerowej odpowiedzialnej za przetrzymywanie i przesyłanie informacji między modułami. Nieodłączną częścią systemu jest także aplikacja webowa pozwalająca za pośrednictwem serwera utrzymywać dwustronną komunikację z telefonem użytkownika.



Schemat połączeń i kierunku wymiany informacji pomiędzy modułami.

Wymagania co do platformy:

- web2py w wersji 1.66.2 lub wyższej
- Python w wersji 2.7.2lub wyższej
- System operacyjny Android wersji 2.2lub wyższej

PROTOKOŁY I ROZWIĄZYWANIE SYTUACJI KRYTYCZNYCH

Protokoły komunikacyjne:

1. System GPS → Smartfon
    - Protokół jednostronnej komunikacji GPS, wymaga widoczności minimum 4 satelitów
    - W przypadku braku widoczności satelitów lub wyłączonej funkcji GPS, system Android potrafi ustalić z mniejszą dokładnością pozycję na podstawie widoczności access-pointów sieci Wi-Fi
2. Smartfon ↔ Serwer
    - Komunikacja HTTP – informacje przesyłane zgodne z typem MIME: plain/text lub multipart jeśli wyniknie konieczność wprowadzenia przesyłu obiektu wieloczęściowego
    - W celu utrzymania wysokiej kultury protokołu komunikacyjnego mechanizm obsługi wyjątków, w przypadku utraty połączenia, odebrania złych danych itp., zapewni wyświetlenie informacji o błędzie w przyjaznej użytkownikowi formie (Android toast)
3. Serwer ↔ Aplikacja webowa
    - Komunikacja HTTP – ze względu na fakt, że aplikacja webowa jest tak naprawdę zwykłą stroną internetową, w przypadku problemów system standardowych komunikatów o błędach zwróci odpowiedni kod (404, 502) lub zapewniony zostanie własny mechanizm informowania o błędach.
    - Jeśli problem wyniknie w części serwerowej aplikacji webowej, wszystkie komunikaty o błędach otrzymują zgodnie ze specyfikacją web2py odpowiedni ticket błędu, co znacznie upraszcza rozwiązywanie problemów, a jednocześnie nie prowadzi do przechowywania, przetrzymywania informacji niespójnych.

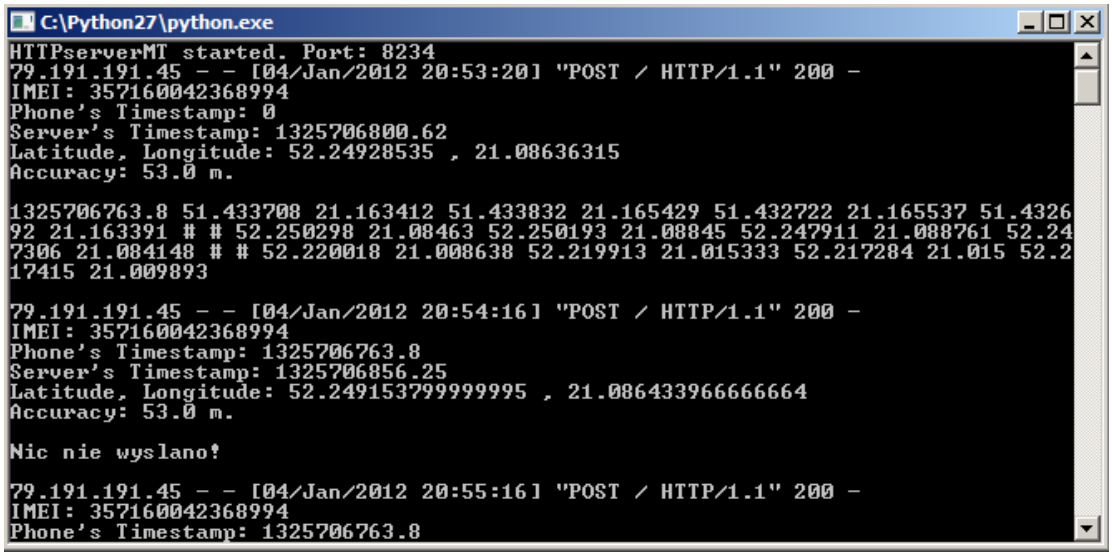
APLIKACJA SMARTFONOWA

Aplikacja składa się z następujących modułów oraz klas:

- *StartingActivity* – startowy obiekt systemu Android uruchamiający inne wątki aplikacji, tworzy kontekst.
- *Network* – moduł odpowiedzialny za komunikacje telefonu z serwerem. Działa w oparciu o protokół HTTP.
- *Data* – moduł zajmujący się obsługa i przechowywaniem danych potrzebnych do sprawdzania położenia użytkownika.
    - *Position* – minimalny obiekt opisujący punkt z użyciem współrzędnych geograficznych;
    - *Zone* – obiekt opisujący dowolny wielokąt, który tworzy strefę w której może przebywać użytkownik, dodatkowo zapewnia metodę sprawdzania czy użytkownik znajduje się dozwolonej strefie.
- *LocationService* –moduł tworzący mechanizmy aplikacji działające w tle systemu. Tworzy wątki (zsynchronizowane w kluczowych miejscach) odpowiedzialne za obsługę:
    - *NotificationManager* – powiadomień użytkownika o jego pozycji oraz o aktualnym stanie aplikacji (*UserLocationState*) :
        - ❖ *OK* – użytkownik znajduje się w jednej z dozwolonych stref;
        - ❖ *BAD* – użytkownik w strefie dozwolonej;
        - ❖ *NOT_KNOWN* – nie jest znany stan użytkownika, np. błędu zarządcy położenia
    - *LocationManager* – zarządca położenia, mechanizm dostarczany przez system Android. Reaguje na zmiany położenia, zarządza: sposobami pozyskiwania położenia (Wi-Fi, nadajników GSM -Base Transceiver Station, sygnału GPS), czasem reakcji zmiany położenia. Zajmuje się zmianą aktualnego stan aplikacji (UserLocationState), na podstawie lokalizacji użytkownika.
- *Log* – klasa usprawniająca logowanie powiadomień niewidocznych dla użytkownika, ułatwia testowanie aplikacji oraz reagowanie na sytuacje wyjątkowe.
  Encrypter – mechanizm deszyfrowania treści otrzymywanej od serwera, zapewnia wysoki poziom bezpieczeństwa, klucz szyfrowania nie jest przesyłany przez kanał.

SERWER

- **main** – startuje aplikację
- **server** – moduł odpowiedzialny za komunikację składa się z 3 części. Serwer jest wielowątkowy i pracuje na porcie 8234.
  Moduły serwera:
  - część odpowiedzialna za odbieranie transmisji od użytkownika systemu Android zapewnia obsługę danych przychodzących – odpowiednio je parsuje i przypisuje do zmiennych. Odpowiada także za prezentacje danych na ekranie konsoli.
  - funkcja komunikacji z modułem web2py
  - odsyłanie danych do użytkownika telefonu – bazując na unikalnym identyfikatorze ważności spisu stref, nowe strefy, lub potwierdzenie ich niezmienności są odsyłane z powrotem do użytkownika telefonu.Dane odsyłane są każdorazowo szyfrowana za pomocą klucza nieprzekazywanego przez kanał zapewniając wysoki poziom bezpieczeństwa transmisji.
- **serverData** – kontener przechowujący dane dotyczące aplikacji: stałe (np. port serwera), jak i zmienne (numery IMEI, poszczególne strefy, przydział stref do danego użytkownika itp.)
- **updater** – zapewnia funkcjonalność odnawiania stanu serwera (dla danego numeru IMEI, aktualny stan dozwolonych stref jest zmieniany)
- **encrypter**–zapewnia wysokie bezpieczeństwo transmisji stosując algorytm szyfrowania (operacja XOR na danych przesyłanych z numerem IMEI)
- Porcje danych przesyłanych:
  - Android → Python:
    `IMEI TIMESTAMP LONGITUDE LATITUDE ACCURACY`
  - Python→ Android:
    `STRING_SIZE(bitowo) ZONE_ID X01 Y01 X02 Y02 X03 Y03 X04 Y04 # # X11 Y11 X12 Y12 X13 Y13(szyfrowane)`
    albo, gdy strefa nie została zmieniona:
    `0`
  - Python → web2py (zapytanie http):
    `"http://localhost:8000/ASA/default/new_position?position="+imei+"?"+coordinates[2]+"?"+coordinates[3]+"?"+coordinates[4]`



WEB2PY

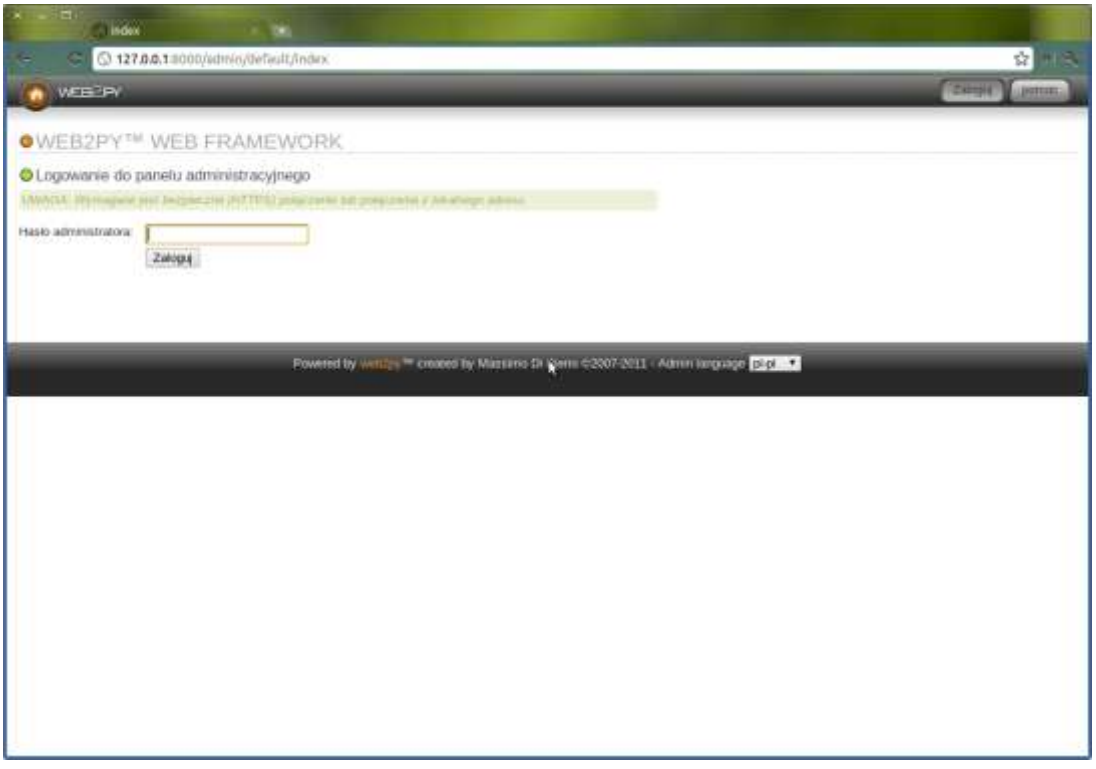
Aplikacja klienta managera (webowa) została utworzona korzystając z frameworka web2py oraz Google Maps API. Framework korzysta z języka Python oraz jest oparty jest o wzorzec projektowy Model-Widok-Kontroler (ang. Model-View-Controller). Zaawansowany mechanizm ticketowania pozwala na łatwe rozwijanie aplikacji oraz śledzenie błędów. Google Maps API korzysta zarówno Pythona jak i JavaScript.

Aplikacja webowa składa się z następujących modułów:

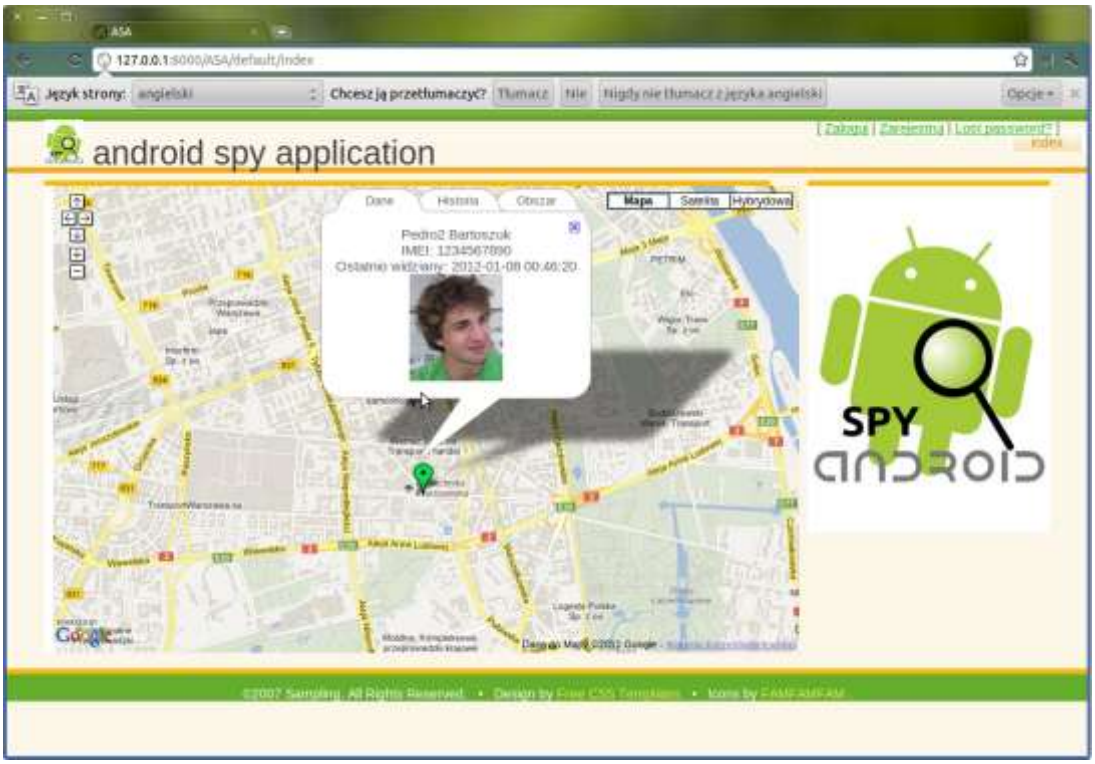
Zarządca bazy danych:

  - korzysta z relacyjnej bazy danych SQLite. Przechowuje w niej dane opisujące użytkowników, historię miejsc które zostały zarejestrowane przez system.
  - Dodatkowo przechowuje uprzednio pobrane współrzędne geograficzne wierzchołków wielokątów, które tworzą strefy w których może przebywać dany użytkownik.

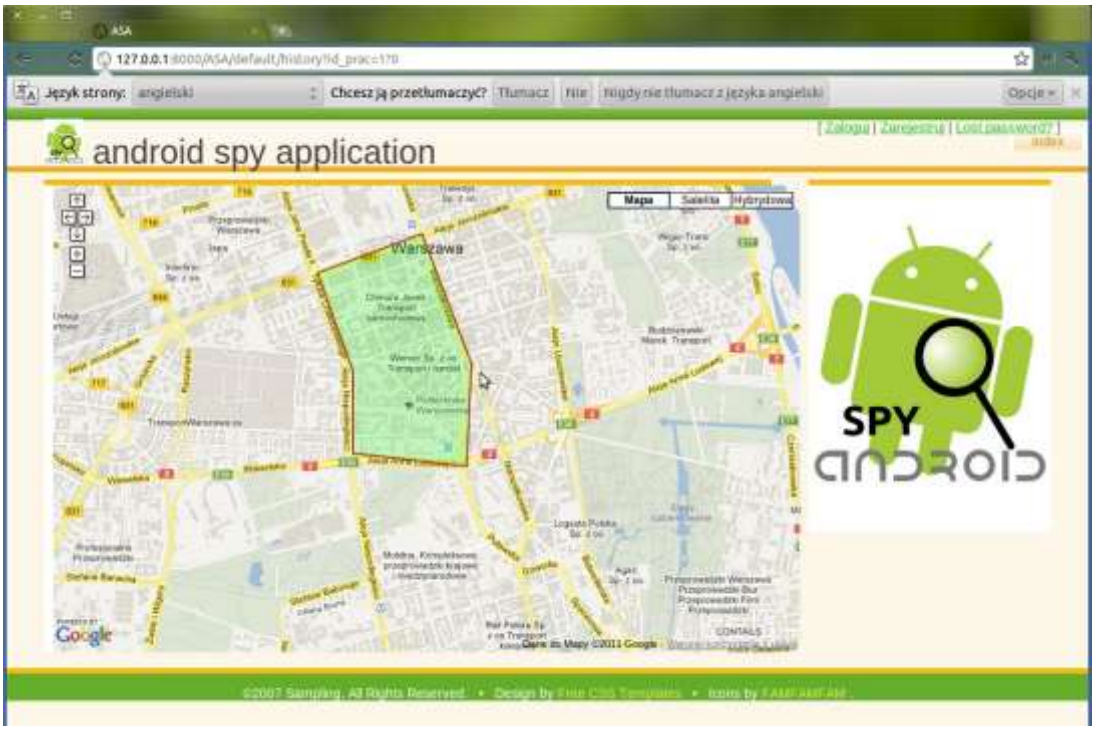
- Google Maps API – zapewnia dobrze znaną prezentacja warstwy danych (stref, pozycji oraz notyfikacji czy dany byt znajduje się wewnątrz czy też na zewnątrz, zawczasu określonej strefy) w przyjaznej dla użytkownika postaci. Za pomocą kolorowych elementów oznacza pozycję użytkownika jak również przypisany mu obszar dozwolony. Po wybraniu danego użytkownika oznaczonego kolorową pinezką mamy wgląd w jego cechy: nazwę, numer IMEI, datę ostatniej obserwacji oraz zyskujemy dostęp do opcji zarządzania. Istnieje możliwość m.in.: zmian dozwolonych obszarów, obejrzenia historii pozycji i ścieżek dla zadanego okresu czasu.
- Panelu administratora – po ewentualnej kompilacji programu, użytkownik ma możliwość autoryzowanego dostępu do bazy danych aplikacji webowej. Ten moduł zapewnia łatwy dostęp do funkcji dodawania nowych pracowników oraz modyfikacji ich cech.



Zrzut ekranu 1. Ekran logowania do aplikacji Managerskiej



Zrzut ekranu 2. Widok pracownika na mapie.



Zrzut ekranu 3. Widok *Zony* dla danego pracownika na mapie.

TESTOWANIE

Charakter aplikacji wymaga testowanie jej w warunkach zbliżonych do naturalnych tj. w miejscu gdzie istnieje widoczność dostatecznej ilości satelitów i gdzie będzie zapewnione połączenie z serwerem. Dla łatwiejszego testowania, zanim zaczniemy etap końcowy na rzeczywistych urządzeniach (większa niepewność pomiarowa, wpływ innych czynników zewnętrznych), wykorzystamy możliwości wirtualnej maszyny Android. Dostarczony przez firmę Google Android Development Kit posiada mechanizm maszyn wirtualnych dla każdej wersji systemu Android. Taka symulacja zapewnia dostęp do wszystkich ustawień, charakterystyki pracy i właściwości systemu Android. Istnieje także, poprzez wtyczkę do programu Eclipse, możliwość dowolnego symulowania położenia poprzez podanie odpowiednich współrzędnych geograficznych (długości i szerokości).

Do testowania aplikacji wykorzystamy zestaw złożony z komputera, telefonu z systemem Android oraz routera Wi-Fi (komunikacja zgodnie z założeniami aplikacji może odbywać się przy wykorzystaniu sieci telefonicznej, ale sieć WLAN znacznie obniżykoszt testów). Aktualnie mamy możliwość sprawdzenia aplikacji pod kątem zgodności z różnymi telefonami i wersjami systemu z jednym z następujących urządzeń:

- Samsung Galaxy S Plus (Android 2.3.3)
- HTC Wildfire (Android 2.2.1)
- HTC Desire (Android 2.2.1).

W celu przetestowania równoległej obsługi wielu użytkowników końcowych przez wielowątkowy serwer, przygotowany zostanie prosty symulator wielowątkowy w Javie (ze względu na duże podobieństwo użytych mechanizmów w odpowiadającej wersji Javy pod Androidem), symulujący dużą liczbę użytkowników, który będzie odpytywał serwer.

Testowanie aplikacji mobilnej wymagało sprawdzenia dokładności danych dostarczanych przez *LocationManagera* (przy czym sama dokładność jest to parametr mierzony przez *LocationManagera*). Testy przeprowadzano dla różnych zagęszczeń nadajników GSM – BTS:

- miasto duże: dokładność do 60 m
- miasto średniej wielkości:100 m
- teren wiejski dokładność w zakresie 500m – 1000m

Dodatkowo włączenie modułu Wi-Fi pozwalało uzyskiwać wyniki ok. 30m w obszarach gdzie sygnał był dostępny.

Bez względu na rozmieszczenie BTS-ów oraz Wi-Fi korzystanie z sygnału GPS dawało wyniki z dokładnością ok. 4m. Jednak w wielu i częstych przypadkach używanie sygnału GPS jest utrudnione ze względu np. ze względu na przebywanie w budynkach. W takich sytuacjach *LocationManager* decyduje z którego providera sygnału lokalizacyjnego należy korzystać. Dodatkowo należało także wybierać czy nowa lokalizacja okazywała się lepsza od starej (zajmuje się tym algorytm zaimplementowany w *LocationService – isBetterLocation()*).

PODZIAŁ PRACY

Podział pracy ze względu na modułowy charakter projektu jest łatwy do ustalenia. Jedynie część dotycząca aplikacji na system Android, ze względu na dużą złożoność i konieczny nakład pracy została podzielona na 2 osoby. Sam przebieg pracy nad aplikacją smartfonową, jak i związany z nią zakres odpowiedzialności będzie modyfikowany na bieżąco w toku prac.

Piotr Jastrzębski:

- Dokumentacja wstępna i końcowa
- Moduł aplikacji smartfonowej: Android – obustronna komunikacja telefon – serwer, wysyłanie współrzędnych i odbieranie zakresów obszarów dozwolonych, interfejs graficzny, część odpowiedzialna za zbieranie danych z czujników i odpowiednie ich przetwarzanie oraz analizowanie.

Piotr Bartoszuk:

- Moduł serwerowy: Python – odpowiednie obsłużenie informacji przychodzących z telefonu oraz od strony aplikacji web2py, aktualizacje stanu przechowywanych danych, komunikacja zwrotna (alert, nowy nieprzekraczalny obszar pobytu, wiadomość).

Piotr Okuła:

- Moduł aplikacji webowej: web2py + API Google Maps – przedstawienie danych o użytkowniku w sposób najbardziej czytelny i jednoznaczny za pomocą API Google Maps. Dodanie możliwości zdalnego ustalania obszaru pobytu użytkownika telefonu i wymuszenie przywołania.

Wojciech Kaczorowski:

- Moduł aplikacji smartfonowej: Android – jak w punkcie pierwszym.
- Dokumentacja końcowa.