ŻYŁA Kamil [1]

Wykorzystanie mechanizmów lokalizacji urządzenia mobilnego w oparciu o Google App Inventor

Inżynieria sterowana modelami (MDE), Google App Inventor, Android, geolokalizacja, urządzenia mobilne

Streszczenie

Inżynieria sterowana modelami zaistniała w wielu dziedzinach wytwarzania oprogramowania, w tym też w dziedzinie tworzenia aplikacji mobilnych. Dzięki niej pojawiła się unikalna możliwość budowania w pełni funkcjonalnych aplikacji na platformę Android bez potrzeby posiadania wykwalifikowanych programistów języka Java.

Celem artykułu jest przedstawienie możliwości budowy aplikacji, wykorzystujących usługi geolokalizacyjne, przy pomocy platformy App Inventor. Oprócz opisu dostępnych narzędzi i procesu tworzenia przykładowej aplikacji, poruszono temat dostępności tak stworzonego oprogramowania.

UTILIZATION OF GEOLOCALIZATION MECHANISMS IN MOBILE APPLICATIONS BASED ON GOOGLE APP INVENTOR

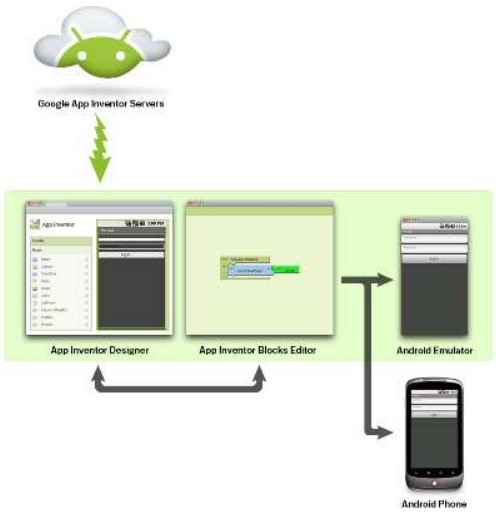
Abstract

Model driven engineering influences on many domains of software development, including applications for mobile devices. It gives unique opportunity to build fully functional software for Android-based devices without participation of IT specialists in such process.

The purpose of this article is to show the abilities of App Inventor platform as the tool for creating Android applications utilizing geolocalization mechanisms. It contains description of the platform, tools and the software creation process.

1. WSTĘP

Inżynieria sterowana modelami zaistniała w wielu dziedzinach wytwarzania oprogramowania, w tym też w dziedzinie tworzenia aplikacji mobilnych. Firma Google Inc. opracowała narzędzie pozwalające w przystępny sposób modelować aplikacje mobilne na platformę Android. Jest to inicjatywa szczególnie istotna dla osób nie posiadających technicznej wiedzy pozwalającej na tworzenie aplikacji mobilnych w środowisku Java. Dzięki niej nie tylko poznają podstawy programowania wizualno-zdarzeniowego, ale również przestają być zwykłymi użytkownikami tych urządzeń. Stwarza to, m.in. przedsiębiorcom, unikalną możliwość budowania (bez potrzeby posiadania wykwalifikowanych programistów języka Java) własnych aplikacji wykorzystujących zdolności lokalizacyjne urządzeń mobilnych Android.



Rys.1. Architektura platformy App Inventor [1]

[1] Politechnika Lubelska, WEiI, Instytut Informatyki, ul. Nadbystrzycka 36 b, 20-618 Lublin, Polska, tel. +(48-81) 525 20 46, fax: +(48-81) 538 43 49, e-mail: kamilz@cs.pollub.pl

Środowisko składa się z dwóch zasadniczych części – projektanta interfejsu i składowych aplikacji (App Inventor Designer) oraz projektanta logiki biznesowej aplikacji (App Inventor Blocks Editor). Pierwszy z nich funkcjonuje na zewnętrznym serwerze, drugi jest aplikacją działającą na lokalnym komputerze. Zgodnie z duchem inżynierii sterowanej modelami, model aplikacji zbudowany przy pomocy projektantów jest podstawą do wygenerowania kodu specyficznego dla platformy. Aplikacje wygenerowane w ramach platformy App Inventor mogą być m.in.:

- automatycznie uruchomione na urządzeniu z systemem operacyjnym Android,
- zainstalowane na urządzeniu z systemem operacyjnym Android,
- zapisane w postaci przenoszalnego instalatora aplikacji.

Ponadto do pracy z App Inventor może służyć rzeczywiste urządzenie Android lub urządzenie wirtualne dostarczone z instalatorem App Inventor lub z Android SDK (rys. 1).

Powyższe, dostępność wielu komponentów oraz plany otwarcia kodu źródłowego [2] sprawiają, że warto przedstawić możliwości platformy App Inventor jako narzędzia do budowy oprogramowania wykorzystującego geolokację.

2. MECHANIZMY APP INVENTOR WYKORZYSTYWANE DO GEOLOKALIZACJI

Za zdolność do lokalizacji w środowisku App Inventor odpowiada komponent LocationSensor z palety Sensors. Lokalizacja może być ustalona na podstawie wskazań odbiornika GPS lub alternatywnych metod takich, jak lokalizacja nadajników operatora telefonii komórkowej lub sieci bezprzewodowych. Komponent dostarcza informacje o długości i szerokości geograficznej, wysokości (jeśli wspierane przez urządzenie) oraz adresie powiązanym ze współrzędnymi. Możliwa jest również konwersja adresu na współrzędne geograficzne. Do poprawnego funkcjonowania komponentu potrzeba ustawienia jego właściwości Enabled na true oraz aktywowanie dla urządzenia ustalania lokalizacji na podstawie wskazań GPS lub wspomnianych metod alternatywnych [3].

Najwazniejsze atrybuty komponentu LocationSensor, to [3]:

1. Accuracy – dokładność w metrach.
2. Altitude – wysokość, na jakiej znajduje się urządzenie.
3. AvailableProviders – lista dostawców usług, np. GPS lub sieci.
4. CurrentAddress – bieżący adres.
5. Latitude – szerokość geograficzna.
6. Longitude – długość geograficzna.
7. ProviderLocked – urządzenie nie zmieni dostawcy usług.
8. ProviderName – nazwa bieżącego dostawcy usług.

Zdarzenia zgłaszane przez komponent, to [3]:

1. LocationChanged(number latitude, number longitude, number altitude) – zgłaszane, gdy urządzenie zgłasza nową lokalizację.
2. StatusChanged(text provider, text status) – zgłaszane, gdy zmienia się status dostawcy usług.

Komponent posiada następujące metody[3]:

1. LatitudeFromAddress(text locationName) – określa szerokość geograficzną na podstawie adresu.
2. LongitudeFromAddress(text locationName) – określa długość geograficzną na podstawie adresu.

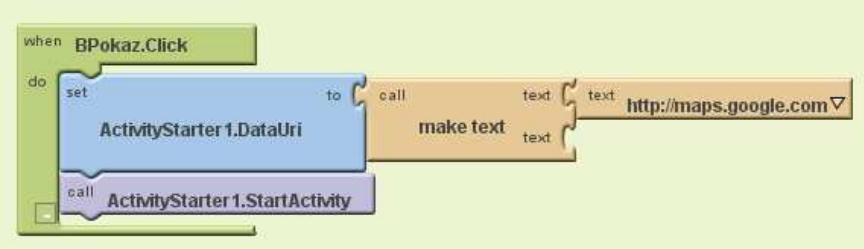
W celu testowania działania aplikacji potrzebne są zmiany współrzędnych zgłaszanych przez urządzenie. W przypadku rzeczywistego urządzenia Android można skorzystać z odbiornika GPS lub ułatwień dla programistów (dostępne w zależności od urządzenia). Kolejną możliwością, działającą również dla emulatora urządzenia Android, jest przesłanie współrzędnych geograficznych przy pomocy polecenia geo fix i programu telnet, np. [4]:

C:\>telnet localhost 5554
Android Console: type 'help' for a list of commands
OK
geo fix -82.411629 28.054553
OK

Kluczowe dane dla określenia lokalizacji, to szerokość (atrybut GPSLatitude) i długość geograficzna (atrybut GPSLongitude). Wartości te są zazwyczaj podawane w stopniach, minutach i sekundach, chociaż zdarza się, że są od razu przeliczone do systemu dziesiętnego, akceptowanego m.in. przez Google Maps. Jeśli wystąpi konieczność przeliczenia współrzędnych, należy pamiętać, że 1 stopień, to 60 minut, a 1 minuta, to 60 sekund. Do obliczenia każdej współrzędnej można wykorzystać następujący wzór: *współrzędnaDzieś=stopnie+minuty/60+sekundy/(60*60)*.

Do uruchamiania zewnętrznych aplikacji można użyć komponentu ActivityStarter z palety Other stuff. Jego działanie polega na wywołaniu aktywności wykorzystując metodę StartActivity (rys. 2). Aby użyć komponentu ActivityStarter w celu wskazania lokalizacji na mapie, należy ustawić następujące właściwości [5]:

1. Action – android.intent.action.VIEW
2. ActivityPackage – com.google.android.apps.maps
3. ActivityClass – com.google.android.maps.MapsActivity
4. DataUri – zawiera informacje o miejscu do wskazania lub informacje o trasie do wyznaczenia. URI powinno być zgodne z formatem Google Maps.



Rys.2. Model uruchamiający aplikację Google Maps

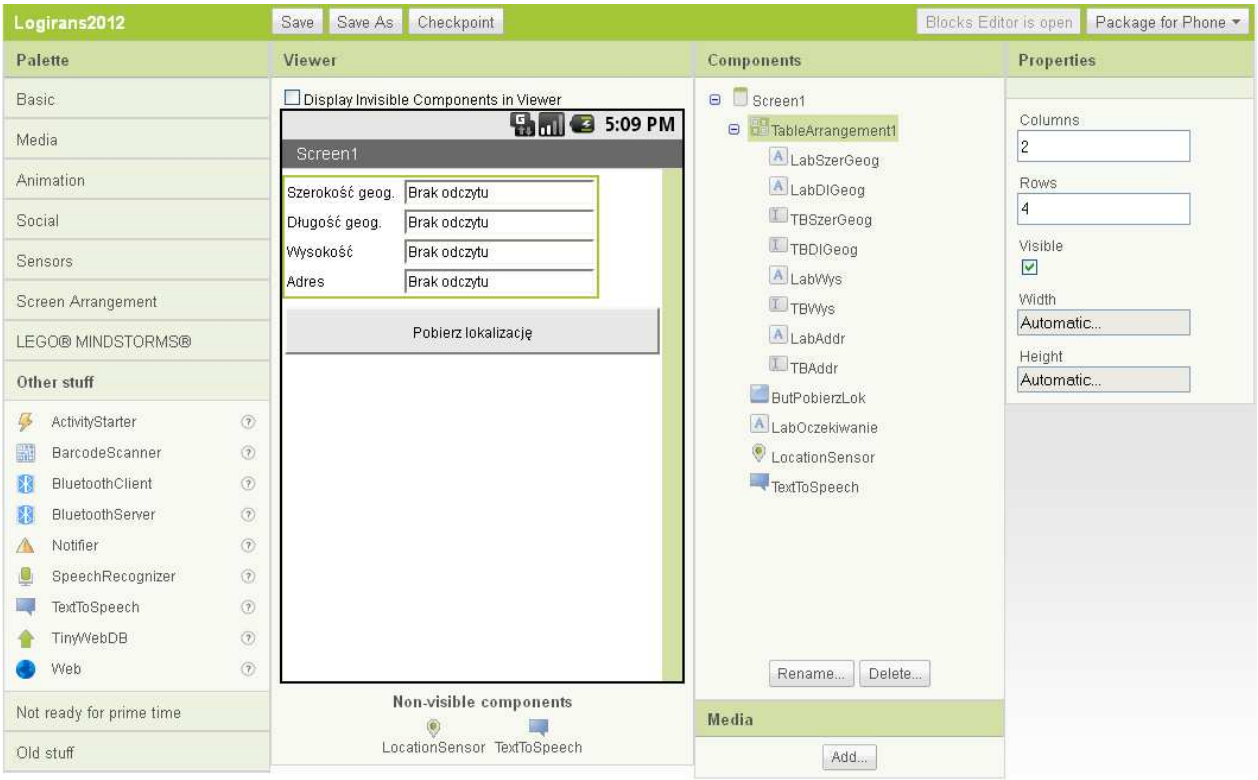
3. ŚRODOWISKO TWORZENIA APLIKACJI

Proces tworzenia aplikacji polega na wykonaniu modelu w App Inventor Designer. Uruchomienie App Inventor Blocks Editor powoduje import i ciągłą synchronizację elementów pomiędzy modelami. Umożliwia również definiowanie logiki działania aplikacji na podstawie metod i atrybutów synchronizowanych elementów oraz metod i atrybutów, które są od nich niezależne. Z poziomu Blocks Editor następuje też połączenie z urządzeniem Android, w celu umieszczenia na nim gotowej aplikacji.

App Inventor Designer udostępnia zestaw komponentów widzialnych na ekranie (elementy interfejsu użytkownika) oraz niewidzialnych (elementy wykonujące pewne akcje). W jego ramach można wyróżnić następujące obszary (rys. 3) [1]:

1. Paleta komponentów – komponenty aplikacji podzielone m.in. na grupy: Basic, Media, Animation, Social, Sensors, Screen Arrangement, LEGO ® MINDSTORMS®.
2. Podgląd interfejsu – podgląd ułożenia komponentów aplikacji.
3. Komponenty – lista komponentów użytych do zbudowania aplikacji.
4. Właściwości – właściwości wybranego komponentu.

Dla przykładu na rysunku 3 przedstawiono aplikację, która po wciśnięciu przycisku Pobierz lokalizację odczytuje adres związany z bieżącą pozycją GPS. Na aplikację składają się elementy interfejsu aplikacji widoczne na ekranie (paleta Basic): etykiety, pola tekstowe wyświetlające współrzędne geograficzne, wysokość oraz adres, przycisk służący do pobrania lokalizacji. Do elementów interfejsu aplikacji niewidzialnych na ekranie zalicza się komponent: LocationSensor odpowiadający za lokalizację (paleta Sensors) oraz TextToSpeach odczytujący na głos adres powiązany ze współrzędnymi (paleta Other stuff). Dodano również zarządcę wyglądu (paleta Screen Arrangement), który umożliwia układ tabelaryczny etykiet i pól tekstowych.

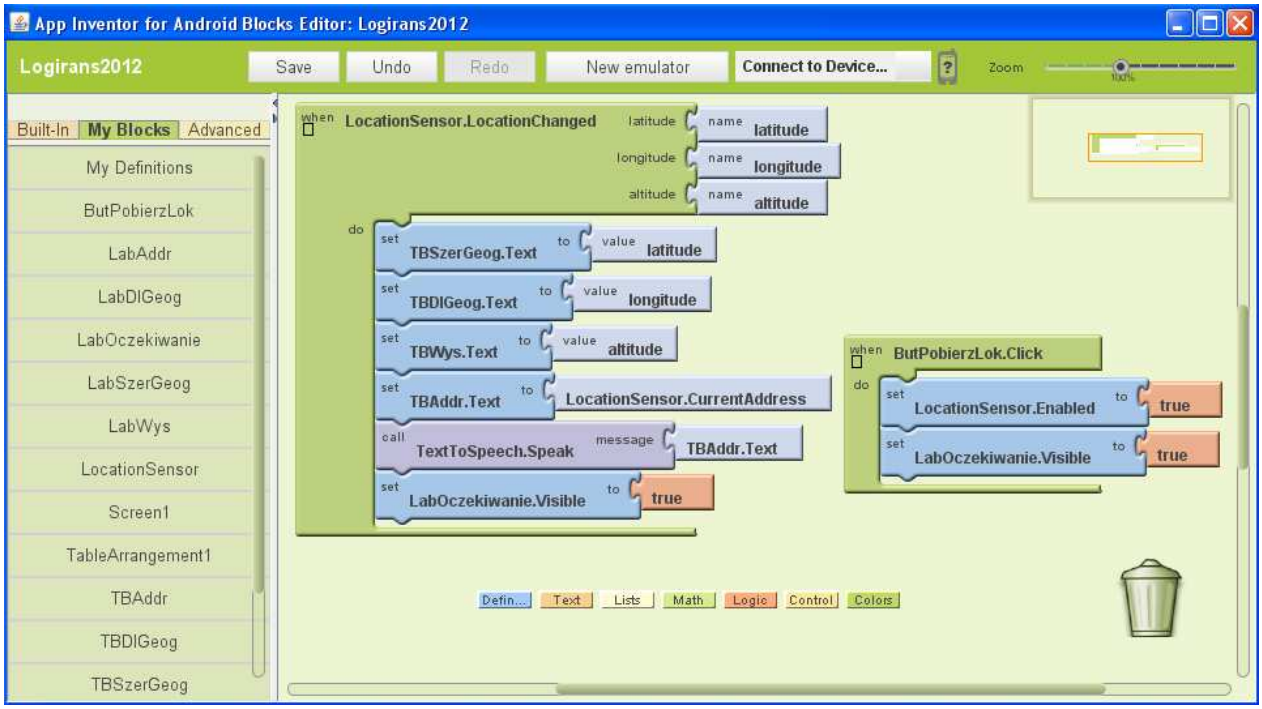


Rys.3. Widok App Inventor Designer

Do modelowania zachowania aplikacji przy pomocy puzzli (symbolizujących zdarzenia i operacje na komponentach) służy App Inventor Blocks Editor, którego podstawowe dwa obszary, to [1]:

1. Paleta komponentów – paleta komponentów służących do definiowania zachowania komponentów interfejsu użytkownika i aplikacji.
2. Podgląd – podgląd modelu przedstawiającego zachowanie aplikacji.

Na rysunku 4 przedstawiono model zachowania aplikacji z rysunku 3. W odpowiedzi na zdarzenie wciśnięcia przycisku zostaje aktywowany nasłuch współrzędnych (komponent true pochodzi z palety Built-in -> Logic). W odpowiedzi na zdarzenie LocationChanged pola tekstowe są wypełniane danymi na podstawie odczytu z LocationSensor, a adres powiązany ze współrzędnymi jest odczytywany. Ponadto etykieta LabOczekiwanie staje się widoczna od momentu wciśnięcia przycisku do momentu uzupełnienia pól tekstowych.



Rys.4. Widok App Inventor Blocks Editor

4. GŁÓWNE MECHANIZMY SKŁADOWANIA DANYCH W APP INVENTOR

Składowanie danych przy pomocy komponentu TinyDB polega na zapisaniu pewnej informacji opatrzonej pewną etykietą przy pomocy metody StoreValue. Etykieta musi być łańcuchem znaków, informacja może być łańcuchem znaków lub listą. Pobieranie informacji z magazynu danych polega na wywołaniu metody GetValue przyjmującej jako argument etykietę, pod jaką zapisano informację. Jeśli pod wskazaną etykietą nie zapisano informacji, zostanie zwrócony pusty łańcuch znaków. Magazyn danych można wyczyścić wybierając z menu telefonu „Settings -> Applications -> Manage Applications -> Clear Data". Każda aplikacja może posiadać tylko jeden magazyn danych (bazę danych) typu TinyDB. Nawet, jeśli w projekcie umieszczono wiele takich komponentów, to i tak odnoszą się do tego samego magazynu danych. Nie można używać komponentu TinyDB do przekazywania danych pomiędzy dwoma różnymi aplikacjami w telefonie. [6]

Z punktu widzenia projektanta aplikacji, składowanie danych przy pomocy komponentu TinyWebDB opiera się na podobnych zasadach, co w przypadku TinyDB. Zasadnicza różnica polega na tym, że TinyWebDB komunikuje się z web service, w celu składowania i pozyskiwania danych, dzięki czemu dane mogą być wymieniane pomiędzy wieloma aplikacjami. Do wykorzystania TinyWebDB jest niezbędny zdalny host z serwisem zarządzającym składowaniem danych. Domyślnie aplikacje łączą się z jego limitowaną wersją udostępnioną, przez firmę Google, pod adresem http://appinvtinywebdb.appspot.com/. Niestety jej główne wady, to ograniczenie liczby wpisów, ograniczenie długości wpisów oraz fakt, że składowane dane są publicznie widoczne i nadpisywane przez nowsze pozycje. Na szczęście każdy może uruchomić swój własny serwis (np. na domowym komputerze) pozbawiony przynajmniej częściowo wymienionych wad. [7]

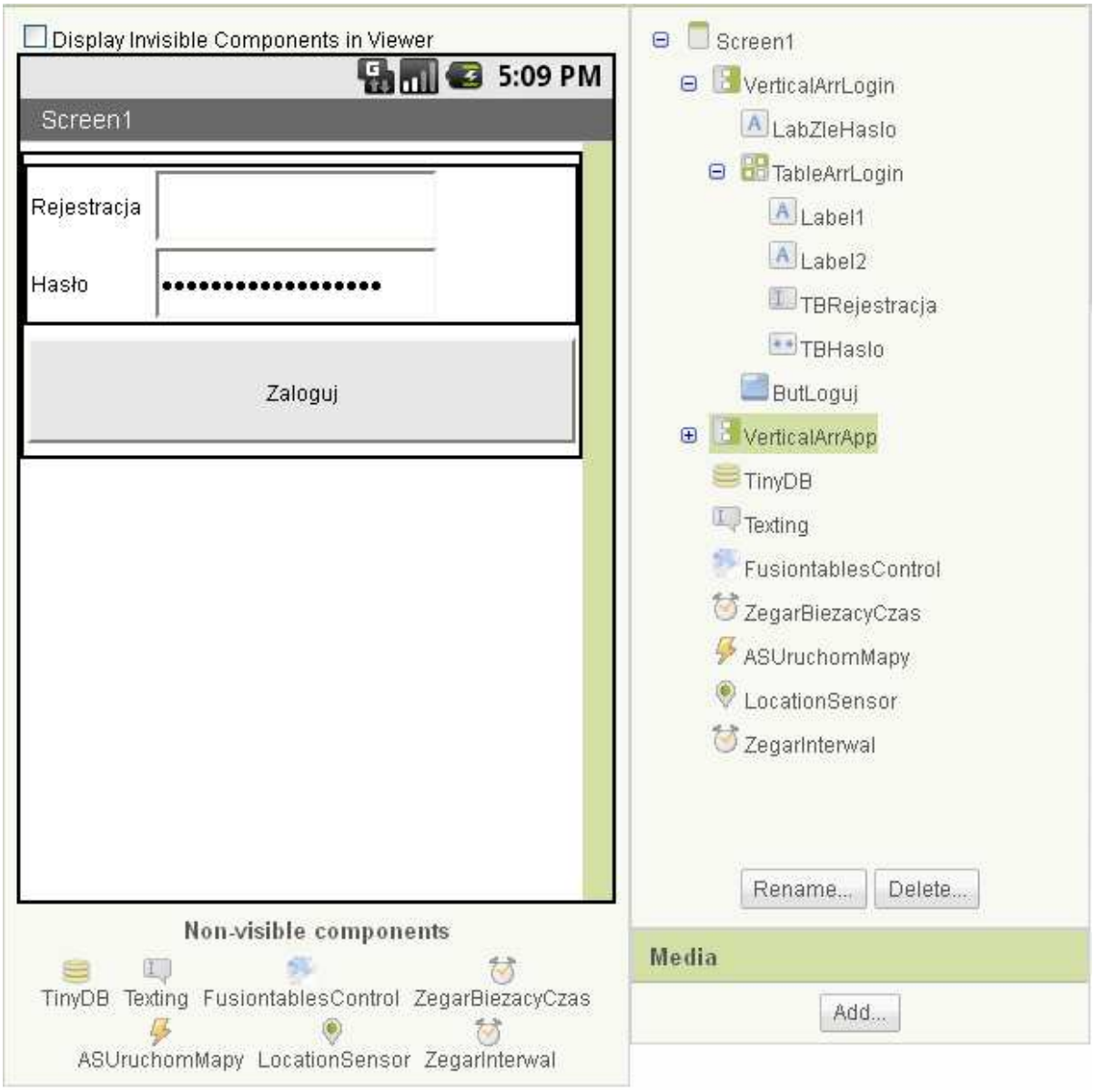
Google Fusion Tables to działająca w chmurze baza danych dostępna dla posiadaczy konta Google (ustawienie tabel jako publicznych pozwala na operowanie na nich bez względu na użyte konto). Pozwala na składowanie, współdzielenie, przeszukiwanie oraz wizualizację danych, a także zarządzanie ich strukturą przy pomocy Fusion Tables SQL API. Bazą danych można zarządzać również przy pomocy interfejsu dostępnego przez przeglądarkę stron internetowych [8]. Komponent środowiska App Inventor współpracujący z tą usługą nazywa się FusiontablesControl, a idea jego działania polega na skonstruowaniu zapytania - atrybut Query (konstruowane zapytanie musi być zgodne ze składnią fusion tables

oraz uwzględniać wielkość liter) oraz wywołaniu metody DoQuery, która je wykona. Zwrócenie rezultatów zapytania (w formacie CSV) powoduje aktywowanie zdarzenia GotResult, w ramach którego można przetwarzać zwrócone dane. Dane z formatu CSV można przekształcić do listy przy pomocy komponentów list from csv table lub list from csv row z palety Lists App Inventor Blocks Editor. [9]

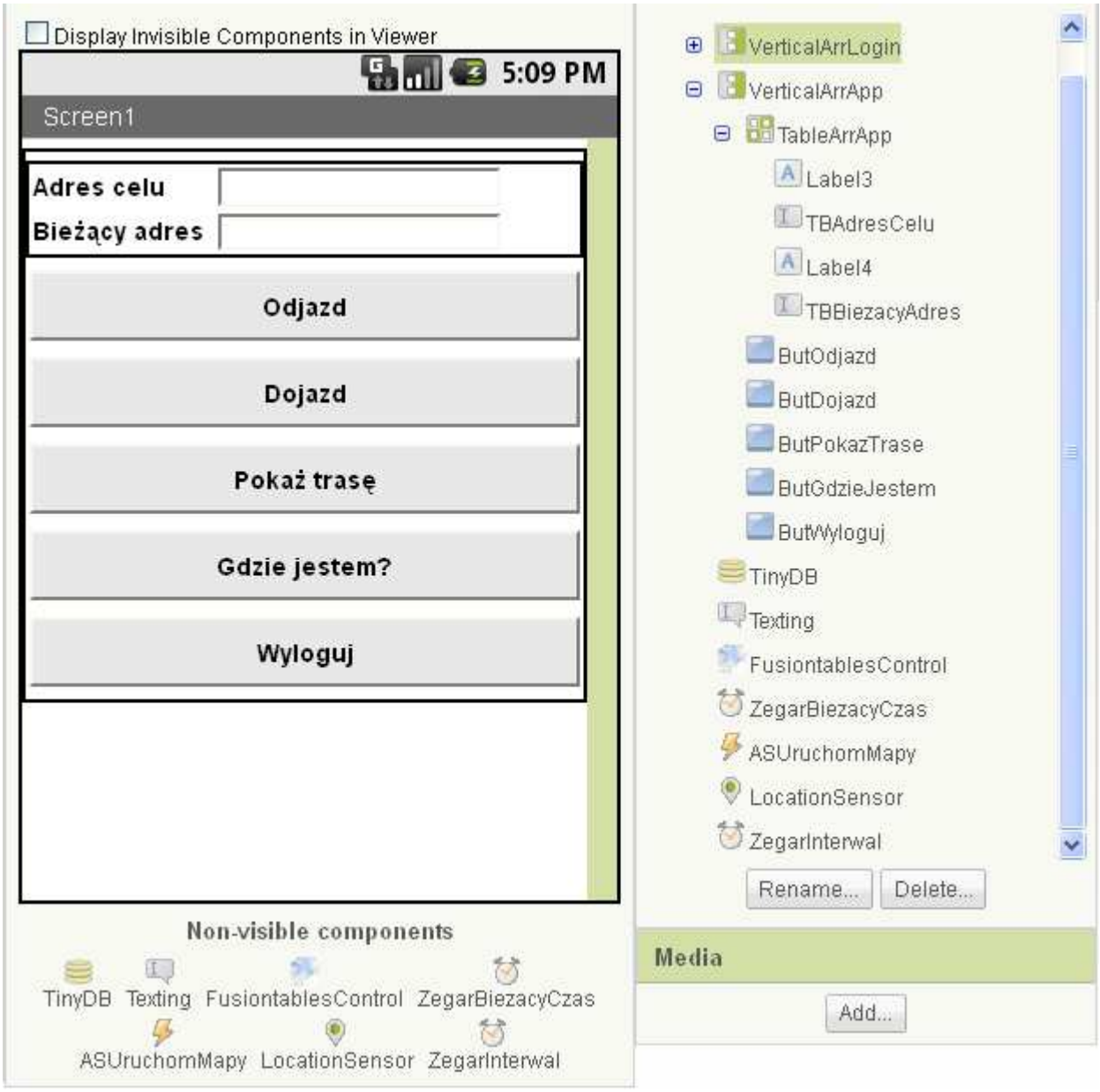
5. PRZYKŁAD WYKORZYSTANIA GEOLOKALIZACJI DO RAPORTOWANIA POŁOŻENIA POJAZDU

Niech celem będzie stworzenie aplikacji, do której loguje się kierowca ciężarówki (podając nr rejestracyjny pojazdu i hasło), po czym wpisuje adres punktu docelowego, do którego będzie podążał i wciska przycisk Odjazd. Chwilę potem do nadzorcy (np. właściciela firmy spedycyjnej lub jej pracownika) jest wysyłany SMS z wybranym punktem docelowym, godziną odjazdu i miejscem odjazdu. Ponadto co np. 10 minut w dostępnej dla nadzorcy zewnętrznej bazie danych (np. fusion tables) pojawia się wpis o zmianie lokalizacji pojazdu. Wciśnięcie przycisku Dojazd wysyła informację o dotarciu do celu i kończy zapisywanie zmian lokalizacji do bazy danych.

W celu realizacji zdefiniowanej funkcjonalności, w App Inventor Designer należy zbudować model interfejsu aplikacji przedstawiony na rysunku 5 i 6. Widzialne elementy interfejsu podzielono na dwa obszary w ramach jednego ekranu: VerticalArrLogin (rys. 5) i VerticalArrApp (rys. 6). W zależności, czy użytkownik jest zalogowany jest widoczny tylko jeden z nich. Zmiana wyświetlanych obszarów zależy od wartości zmiennej logicznej. Ponadto aplikacji towarzyszy zbiór komponentów niewidzialnych na ekranie, w postaci komponentów odpowiedzialnych za: pozyskiwanie czasu urządzenia (ZegarBiezacyCzas i ZegarInterwal), pozyskiwanie lokalizacji urządzenia (LocationSensor), uruchomienie aplikacji Google Maps (ASUruchomMapy), wysyłanie SMS (Texting), trwałe składowanie danych, w tym ustawień aplikacji, w ramach aplikacji (TinyDB) oraz trwałe składowanie danych o zmianach lokalizacji w Fusion tables (FusiontablesControl).



Rys.5. Widok logowania do aplikacji



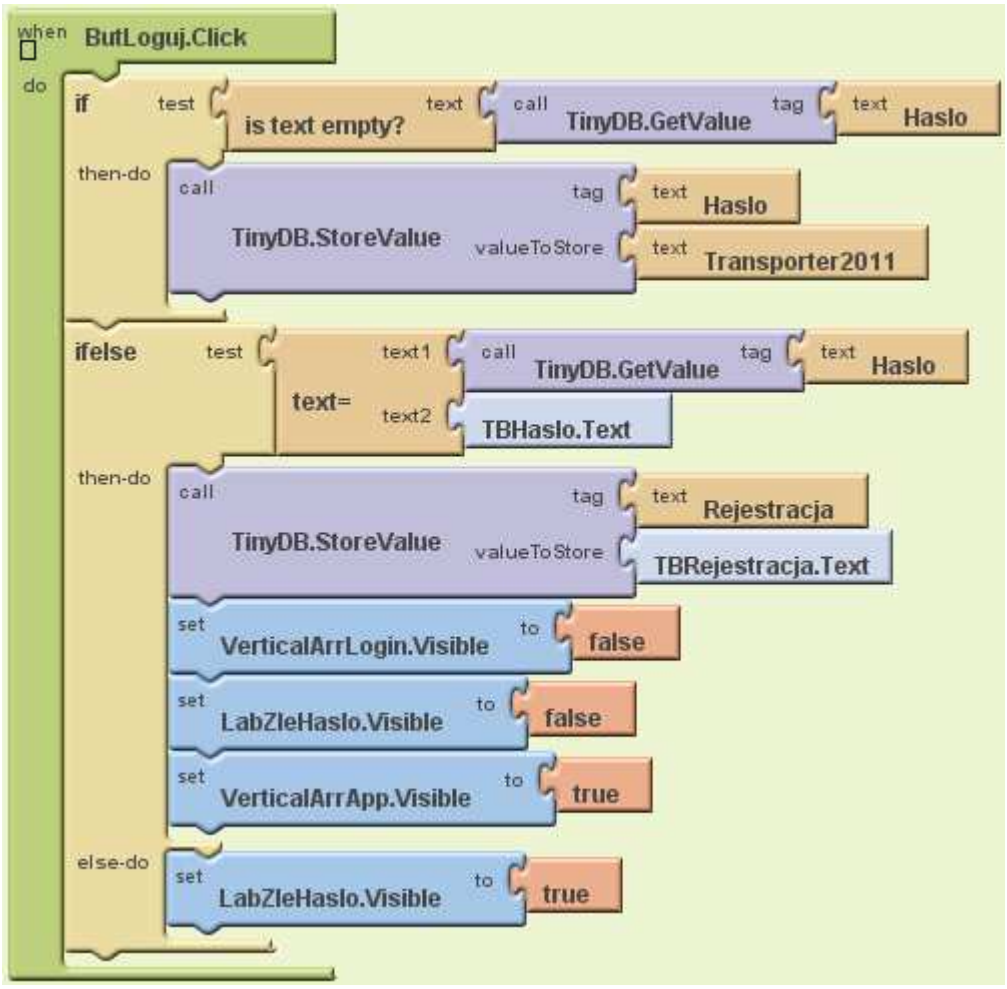
Rys.6. Widok funkcjonalności aplikacji

Funkcjonalność aplikacji (rys. 6) jest następująca:

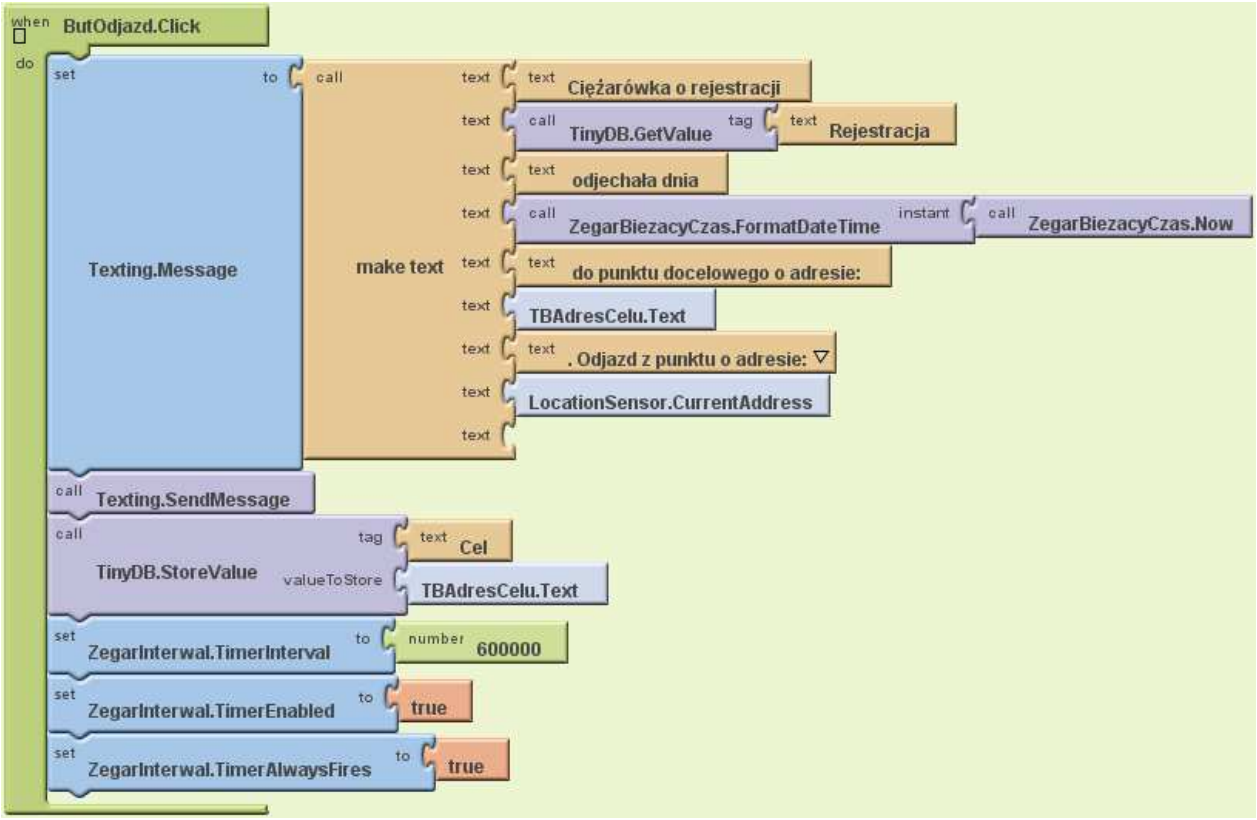
1. Adres celu – pole tekstowe, w które kierowca wprowadza adres miejsca docelowego.
2. Bieżący adres – pole tekstowe, aktualizowane wraz ze zmianą położenia pojazdu, zawierające adres związany z aktualną pozycją.
3. Odjazd – przycisk, którego wciśnięcie rozpoczyna monitorowanie pozycji pojazdu.
4. Dojazd – przycisk, którego wciśnięcie informuje o końcu trasy. Monitorowanie położenia pojazdu ulega zakończeniu.
5. Pokaż trasę – przycisk uruchamiający Google Maps w trybie nawigacji.
6. Gdzie jestem? – przycisk uruchamiający Google Maps w trybie oznaczenia bieżącej pozycji.
7. Wyloguj – przycisk, który umożliwia przejście do widoku logowania – osoby niepowołane nie mają dostępu do funkcjonalności aplikacji.

Kolejnym etapem jest przypisanie określonej funkcjonalności i zdarzeń do elementów interfejsu użytkownika. Na rysunku 7 przedstawiono realizację logowania do aplikacji. W odpowiedzi na wciśnięcie przycisku Zaloguj następuje sprawdzenie, czy zdefiniowano hasło – jeśli nie, jest dodawany odpowiedni wpis w bazie danych. Kolejne sprawdzenie dotyczy porównania hasła podanego przez użytkownika z hasłem zapisanym w bazie danych. Jeśli stwierdzono zgodność haseł, jest wyświetlany widok funkcjonalności aplikacji, a chowany widok logowania.

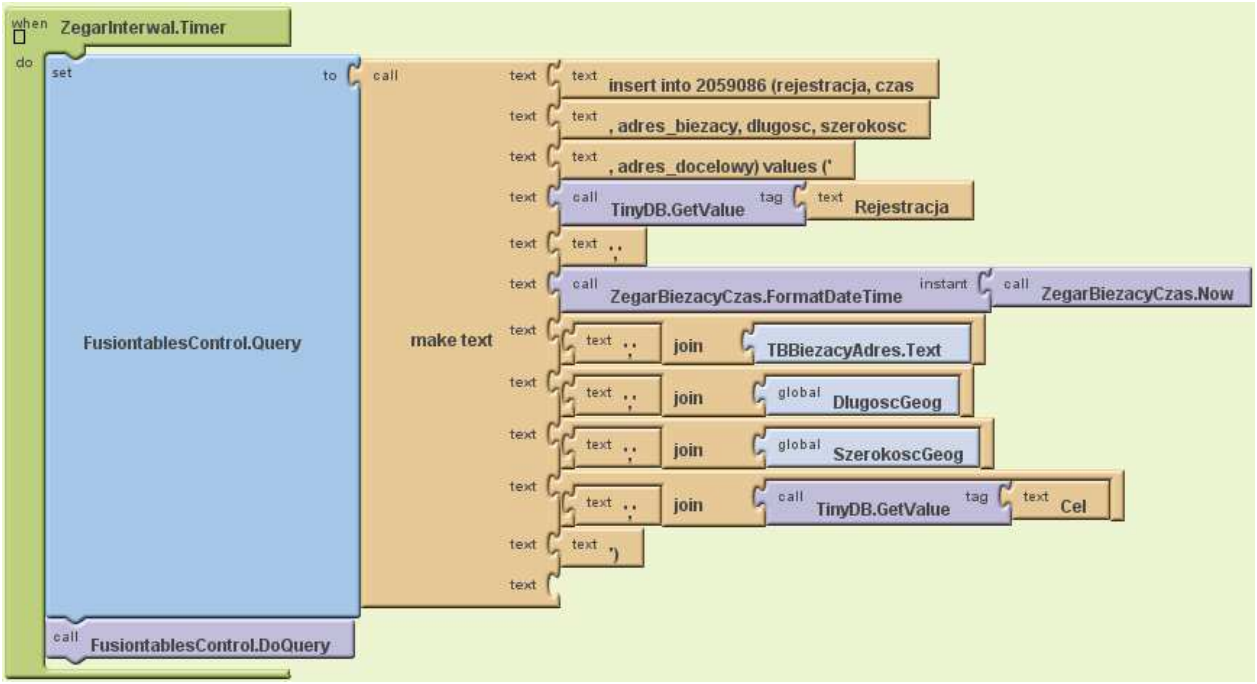
Na rysunku 8 i 9 przedstawiono reakcję aplikacji na odjazd pojazdu oraz zbieranie tzw. „okruszków", czyli kolejnych pozycji pojazdu. Po wciśnięciu przycisku Odjazd jest: komponowany i wysyłany SMS zawierający dane transportu, zapisywany w bazie danych typu TinyDB cel transportu oraz ustawiany zegar zapewniający zbieranie „okruszków". Zdarzenie ZegarInterwal.Timer zapisuje, w zdefiniowanych odstępach czasu, w Fusion tables lokalizację pojazdu.



Rys.7. Obsługa zdarzenie wciśnięcia przycisku Loguj

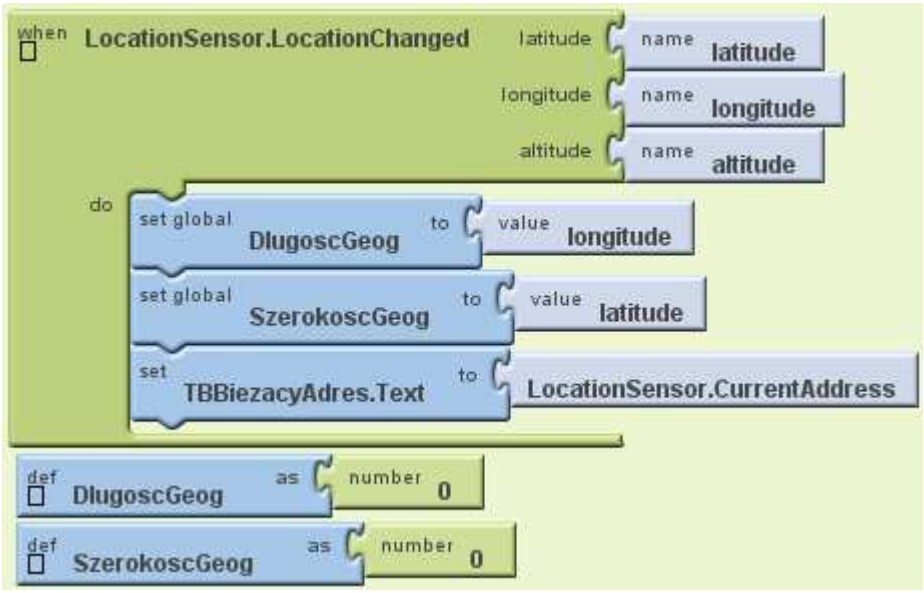


Rys.8. Obsługa zdarzenia wciśnięcia przycisku Odjazd



Rys.9. Model zdarzenia zegara zapisującego „okruszki" w Google Fusion tables

Model przedstawiony na rysunku 10 jest najważniejszym elementem zachowania aplikacji – rejestruje zmiany położenia pojazdu. Zdarzenie LocationChanged jest wyzwalane w momencie zmiany położenia urządzenia mobilnego, a tym samym pojazdu. W jego ramach do zmiennych przechowujących lokalizację są zapisywane nowe współrzędne, a następnie w polu Bieżący adres pojawia się adres wyznaczony na podstawie nowych współrzędnych. Zapamiętane wartości są wykorzystywane w innych częściach aplikacji.



Rys.10. Obsługa zdarzenia polegającego na zmianie lokalizacji urządzenia mobilnego

6. WNIOSKI

Modele przedstawione w niniejszym artykule pokazują, że stworzenie relatywnie skomplikowanej aplikacji nie wymaga szczególnie wiele wysiłku. Napisanie jej od podstaw zajęłoby więcej czasu oraz dodatkowo wymagało umiejętności programowania w języku Java i znajomości platformy Android. Dodatkowo wykorzystanie fusion tables umożliwia dalsze przetwarzanie danych przy pomocy dowolnego innego oprogramowania.

Modele są na tyle proste, że niewiele więcej problemów sprawiłoby rozszerzenie aplikacji o dodatkową funkcjonalność – np. personalizację, edycję numeru telefonu nadzorcy, rozbudowanie zakresu raportowanych parametrów. Ponadto przykładowa aplikacja, po niewielkich zmianach, może służyć jako podstawa dla całej klasy podobnych aplikacji np. pomagającej starszym ludziom (i nie tylko) dotrzeć do domu lub powiadomić opiekuna o ich położeniu.

Implementację mechanizmów geolokalizacyjnych w Google App Inventor można uznać za wystarczającą, jednakże jako platforma do modelowania aplikacji mobilnych, posiada on pewne ograniczenia. Są to przede wszystkim: ograniczona liczba komponentów, relatywnie niewielkie możliwości kształtowania wyglądu interfejsu użytkownika aplikacji oraz brak oficjalnego wsparcia dla „pełnoprawnych" baz danych np. sqlite lub MySQL. Niemniej ograniczenie do narzędzi dostarczonych przez twórców platformy jest poniekąd nieuniknioną ceną za prostotę tworzenia oprogramowania, a przez to poszerzenie kręgu jego twórców o osoby spoza grona specjalistów/programistów.

Warto również zwrócić uwagę na to, że: opiekę nad platformą App Inventor sprawuje obecnie Massachusetts Institute of Technology, uczelnie wyższe (m.in. Politechnika Lubelska i Uniwersytet w San Francisco) wychodząc naprzeciw dynamicznym zmianom technologii używają platformy w ramach prowadzonych kursów, a także pojawiają się głosy, że App Inventor jest doskonałym narzędziem do nauki myślenia algorytmicznego wśród studentów stykających się po raz pierwszy z programowaniem.

Wszystko to oraz otwartość platformy App Inventor zwiastują pojawienie się nowych komponentów i rozwiązań, więc warto monitorować zmiany, jakie w niej zachodzą.

7. BIBLIOGRAFIA

[1] http://www.appinventorbeta.com: pierwotna strona domowa platformy App Inventor, dostęp - grudzień 2011
[2] http://appinventoredu.mit.edu: nowa strona domowa platformy App Inventor, dostęp - grudzień 2011
[3] http://www.appinventorbeta.com/learn/reference/other/locationsensor.html: dokumentacja komponentu LocationSensor, dostęp - grudzień 2011
[4] http://developer.android.com/index.html: strona dla twórców oprogramowania na platformę Android, dostęp - grudzień 2011
[5] http://www.appinventorbeta.com/learn/reference/other/activitystarter.html: dokumentacja komponentu ActivityStarter, dostęp - grudzień 2011
[6] http://www.appinventorbeta.com/learn/reference/components/basic.html#TinyDB: dokumentacja komponentu TinyDB, dostęp - grudzień 2011
[7] http://www.appinventorbeta.com/learn/reference/components/other.html#TinyWebDB: dokumentacja komponentu TinyWebDB, dostęp - grudzień 2011
[8] http://www.google.com/fusiontables/public/tour/index.html: strona firmy Google dedykowana Fusion tables, dostęp - grudzień 2011
[9] http://www.appinventorbeta.com/learn/reference/components/notready.html#FusiontablesControl: dokumentacja komponentu FusiontablesControl, dostęp - grudzień 2011