POLITECHNIKA WARSZAWSKA
Wydział Elektroniki i Technik Informacyjnych
Instytut Telekomunikacji

PRACA DYPLOMOWA INŻYNIERSKA

Sebastian Łuczak, Maciej Nowak

System kontroli czasu pracy pracowników mobilnych oparty o technologię zbliżeniową Near Field Communication

Kierujący pracą:
mgr inż. Marcin Golański

.....
ocena pracy

.....
data i podpis Przewodniczącego
Komisji Egzaminacyjnej

Warszawa, wrzesień 2011

Streszczenie

Niniejsza praca ma na celu przedstawienie możliwości wykorzystania technologii komunikacji zbliżeniowej NFC do kontroli czasu pracy pracowników mobilnych. Obok aspektów teoretycznych, została zaprezentowana koncepcja oraz pełna realizacja systemu mającego na celu spełniać powyższe funkcje. Praca porusza problemy doboru technologii, opisuje możliwości wykorzystania systemu oraz dalsze perspektywy jego rozwoju. Zostały także przedstawione wyniki testów funkcjonalnych stworzonego rozwiązania oraz wynikające z nich ograniczenia.

Abstract

The purpose of this thesis is to present the possibility of using NFC proximity communication technology in work-time control systems aimed at mobile workers. Besides theoretical aspects, it presents the concept and full implementation of designed system. It brings up the technology selection problems, describes the use cases of the system and further prospects of its development. The results of functional tests and usage limitations were also described.

Maciej Nowak
Specjalność: Telekomunikacja – Systemy i Sieci Telekomunikacyjne
Data urodzenia: 8 lipca 1988
Data rozpoczęcia studiów: 01 października 2007

Życiorys

Urodziłem się 08.07.1988 r. w Warszawie. Po ukończeniu szkoły podstawowej i gimnazjum, kontynuowałem naukę w XXVIII L.O. im. Jana Kochanowskiego w Warszawie. W szkole średniej uczęszczałem do klasy o profilu matematyczno-fizyczno-informatycznym. W październiku 2007 r. rozpocząłem studia dzienne na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej.

……
Podpis

EGZAMIN DYPLOMOWY
Złożył egzamin dyplomowy w dniu ……
Ogólny wynik studiów …….
Dodatkowe uwagi i wnioski Komisji
……
……
……

Sebastian Łuczak
Specjalność: Telekomunikacja – Systemy i Sieci Telekomunikacyjne
Data urodzenia: 21 czerwca 1988
Data rozpoczęcia studiów: 01 października 2007

Życiorys

Urodziłem się 21.06.1988r. w Warszawie. Po ukończeniu szkoły podstawowej, i gimnazjum, kontynuowałem naukę w XXVIII L.O. im. Jana Kochanowskiego w Warszawie. W szkole średniej uczęszczałem do klasy o profilu matematyczno-fizyczno-informatycznym. W październiku 2007 r. rozpocząłem studia dzienne na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej.

……
Podpis

EGZAMIN DYPLOMOWY
Złożył egzamin dyplomowy w dniu ……
Ogólny wynik studiów …….
Dodatkowe uwagi i wnioski Komisji
……
……
……

Spis Treści

Zawartość nośnika elektronicznego:

- Praca inżynierska w formacie .docx i .pdf
- Kody źródłowe aplikacji mobilnych oraz zbudowane paczki .apk
- Obraz maszyny wirtualnej VirtualBox zawierający serwer bazy danych, serwer aplikacji oraz kody źródłowe aplikacji stacjonarnych

Wykaz używanych skrótów

API – Application Programming Interface
CSS - Cascading Style Sheets
DAO – Data Access Object
EJB – Enterprise Java Beans
HQL – Hibernate Query Language
HTML – HyperText Markup Language
http – HyperText Transfer Protocol
IDE – Integrated Development Environment
IT – Information Technology
J2EE – Java Enterprise Edition
JAX-RS – Java Api for Restfull Web Services
JDBC – Java DataBase Connectivity
JSP – Java Server Pages
JRE – Java Runtime Environment
JSON – JavaScript Object Notation
JSTL – Java Standard Tag Library
JVM – Java Virtual Machine
MVC – Model-View-Controller
NFC – Near Field Communication
ORM – Object-Relational Mapping
POJO – Plain Old Java Object
QBE – Query By Example
REST – Representational State Transfer
RFiD – Radio Frequency Identification
SDK – Software Development Kit
SHA – Secure Hash Algorithm
SQL – Structured Query Language
SOAP – Simple Object Access Protocol
UID – Unique Identifier
URI – Uniform Resource Identifier
WWW – World Wide Web
XML – eXtesible Markup Language

1. WSTĘP

Zagadnieniem o dużym znaczeniu biznesowym jest w dzisiejszych czasach możliwość precyzyjnego rozliczania pracowników z wykonywanych obowiązków. Chociaż stacjonarne rozwiązania ulokowane w budynkach istnieją i funkcjonują już od bardzo wielu lat, to wciąż nierozwiązanym problemem jest możliwość transparentnego monitorowania osób, które wykonują swoją pracę w terenie.

Tematem poniższej pracy jest wykazanie możliwości zbudowania rozproszonego systemu kontroli czasu pracy dla pracowników mobilnych z wykorzystaniem nowych technologii, ze szczególnym uwzględnieniem technologii komunikacji zbliżeniowej Near Field Communication.

NFC jest obecnie jedną z najbardziej popularnych technologii komunikacji zbliżeniowej. W jej dynamiczny rozwój inwestuje dziś wielu globalnych potentatów rynku IT, w tym Google, Nokia, Samsung czy też AT&T. Choć głównym obiektem ich zainteresowania jest realizacja bezstykowych płatności elektronicznych, specyfika tego rozwiązania pozwala na wykorzystanie go w wielu innych dziedzinach życia, w tym kontroli czasu pracy.

2. CEL I ZAKRES PRACY

Celem pracy inżynierskiej było zrealizowanie kompletnego i nowatorskiego systemu kontroli czasu pracy pracowników mobilnych. Innowacyjne wykorzystanie wchodzącej na rynek konsumencki technologii NFC (ang. *Near Field Communication*), telefonii komórkowej oraz usług internetowych czyni opisywane rozwiązanie unikalnym na rynku.

Dzięki użyciu infrastruktury telefonii 3G, oraz złożonego systemu monitorowania uzyskać można pełną i niezawodną kontrolę nad czasem poświęconym przez pracownika na realizację powierzonych mu zadań.

Kompletna realizacja projektu zakłada dostarczenie zestawu aplikacji współpracujących ze sobą, na który składają się: baza danych, aplikacja serwerowa będąca usługą sieciową, internetowy portal informacyjny o przeznaczeniu administracyjnym oraz dwie łączące się z nimi aplikacje mobilne (kliencka i administracyjna) na telefon komórkowy z systemem Android. Każdy z tych elementów zostanie przedstawiony oddzielnie w dalszej części pracy.

Technologie użyte do wykonania pracy są powszechnie wykorzystywane do tworzenia oprogramowania biznesowego przeznaczonego dla dużej ilości jednoczesnych użytkowników i zgodne z powszechnymi trendami realizacji usług w oparciu o sieć Internet.

Prezentowana realizacja ma być w założeniu alternatywą dla istniejących już systemów opartych o architektury stacjonarne, zrealizowane przy pomocy stacjonarnych czytników RFiD (ang. *Radio Frequency Identification*) i kart zbliżeniowych, a także stanowić rozwiązanie problemu kontroli czasu pracy osób których profil zatrudnienia zakłada stałe przemieszczanie się po kraju w celu serwisowania urządzeń bezpośrednio u klienta.

Dotychczasowe systemy ze względu na złożoną, specyficzną i kosztowną infrastrukturę nie odpowiadały potrzebom dużej grupy pracodawców, którzy współpracę z kontrahentami opierają na umowach okresowych (serwis urządzeń, wsparcie techniczne IT itp.), a wynagrodzenie pracowników uzależniają od czasu poświęconego na wykonanie zlecenia. W takiej sytuacji potrzebna jest duża łatwość modyfikacji rozmieszczenia punktów kontrolnych oraz ich bezproblemowa, nie wymagająca żadnych narzędzi i ingerencji w otoczenie, instalacja u kontrahentów. Istotnym jest również zapewnienie transparentności i łatwości użytkowania dla pracowników.

Zarówno koncepcja jak i projekt aplikacji został wymyślony i zrealizowany przez dwóch autorów. Ze względu na modułowość zaprojektowanego rozwiązania, dokonany został podział pracy nad implementacją systemu na dwie części – mobilną i stacjonarną, leżącą po stronie serwera.

Aplikacje przeznaczone na telefon komórkowy stworzył Sebastian Łuczak, z kolei aplikacje usługi internetowej, portalu administracyjnego oraz bazę danych Maciej Nowak. Prace nad interfejsem portalu wykonane były wspólnie.

W dalszej części pracy przedstawione zostaną rozważania na temat doboru poszczególnych technologii, powody zastosowania konkretnych technik implementacji oraz scenariusz użycia kompletnego systemu, wraz z opisem zrealizowanego oprogramowania.

3. KONCEPCJA PRACY

3.1. WSTĘP

W poniższym rozdziale przybliżone zostaną główne założenia oraz budowa zaprojektowanego systemu.

Głównym zadaniem przezeń realizowanym jest ułatwienie nadzoru nad pracownikami mobilnymi, głównie poprzez umożliwienie rzetelnego rozliczenia ich czasu pracy. Celem pobocznym, maksymalne możliwe uproszczenie interfejsu użytkownika tak, aby system nie był uciążliwy dla pracowników mobilnych.

W prezentowanym rozwiązaniu nastąpiło odwrócenie klasycznego scenariusza kontroli czasu pracy, w którym czytnik RFiD ulokowany jest na stałe w ścianie lub bramce wejściowej do obszaru roboczego, a pracownik posiada kartę RFiD.

Przedstawiona sytuacja ma ekonomiczne uzasadnienie jeżeli odnosi się do stałej lokalizacji pracowników i ich dużej liczbie na przykład biur, serwisów stacjonarnych itp. Problemy pojawiają się, gdy lokalizacji jest wiele (tyle ilu klientów) i mogą się zmieniać, pracownicy stale poruszają się pomiędzy nimi, a pracodawcy zależy na informacji na temat jakości obsługi klienta i sumienności swoich pracowników. W takiej sytuacji drogi czytnik musiałby zostać umieszczony w dziesiątkach jak nie setkach punktów, a całość musiałaby być spięta w dodatkową sieć transmitującą dane.

W proponowanym scenariuszu czytnikiem jest telefon, a zatem urządzenie w które w dzisiejszych czasach każdy pracownik mobilny jest wyposażony. Nie ma problemu związanego z dodatkowym kosztem czytnika, gdyż zawiera się on w cenie urządzenia używanego przez pracownika. Drugi sprzętowy element systemu, czyli znaczniki komunikacji zbliżeniowej rozmieszczane na każdym z serwisowanych urządzeń, odznaczają się z kolei niezwykle niską ceną (0.44$ za sztukę przy zakupie 1000-4999 sztuk [1]). Ostatni element generujący dodatkowe koszty, czyli kablowa infrastruktura, zastąpiona jest z kolei wykorzystywaniem gotowej infrastruktury sieci komórkowej oferującej pakietową transmisję danych. Dzięki temu drastycznie obniżone zostają koszty, jednocześnie osiągając efekt niedostępny wcześniej, czyli możliwość precyzyjnego określenia czasu pracy pracowników rozproszonych poza siedzibą firmy.

Dopełnieniem całości systemu jest specjalne oprogramowanie łączące telefon komórkowy z aplikacjami serwerowymi. Dzięki niemu możliwe jest zgłaszanie zleceń oraz ich obsługa. Stworzony portal administracyjny pozwala na bieżące administrowanie systemem oraz wgląd do danych zgromadzonych w bazie danych.

Dane transmitowane są w czasie rzeczywistym, dzięki czemu koordynator pracy zna obecny stan zajętości zasobów ludzkich, a także jest w stanie ustalić z większą dokładnością średni czas realizacji danej usługi serwisowej.

Zauważalną wadą zaproponowanego rozwiązania, jest wrażliwość na dostępność usług transmisji danych. Wprowadzenie możliwości pracy telefonu w trybie offline, w którym zbierałby dane o zdarzeniach do pamięci i wysyłał w momencie uzyskania dostępu do sieci, nie jest jednak odpowiednim rozwiązaniem. Obecnie baza danych przechowuje dla każdego wpisu zarówno znacznik czasowy terminala użytkownika jak i serwera obsługującego zdarzenie, ale przy rozliczeniu brana pod uwagę jest data serwera. W wariancie w którym telefon mógłby zbierać dane o zdarzeniach również poza zasięgiem sieci, datą braną pod uwagę musiałaby być data systemowa telefonu, a nie serwera. Działanie takie stworzyłoby możliwość łatwego oszukiwania logiki biznesowej poprzez ręczną modyfikację daty systemowej, lub celowe emulowanie przebywania poza zasięgiem sieci tak, aby zamaskować opóźnione rozpoczęcie lub zakończenie realizacji zlecenia.

3.2. ZAŁOŻENIA I BUDOWA SYSTEMU

System składa się z dwóch aplikacji stacjonarnych, dwóch mobilnych oraz bazy danych. Aplikacje mobilne komunikują się z usługą internetową za pośrednictwem protokołu HTTP (ang. *HyperText Transfer Protocol*) i poprzez wymianę w formacie JSON (ang. *JavaScript Object Notation*) zapytań i odpowiedzi POST. Dane zgromadzone przez usługę lokowane są w bazie danych, a aplikacja portalu administracyjnego dostarcza możliwość wglądu oraz ich edycji.

Rysunek ilustruje uproszczony schemat systemu:

Rys. 3.1 Uproszczony schemat systemu

Poniżej zostały przedstawione założenia mające istotny wpływ na poprawne funkcjonowanie całego systemu:

- Pracownik jest związany z telefonem i kartą SIM – w bazie danych przechowywane są informacje takie jak numer IMEI oraz IMSI urządzenia którym posługuję się pracownik. Pozwala to jednoznacznie zweryfikować tożsamość osoby z telefonu której przyszło zgłoszenie.

- Pracownik może obsługiwać tylko jedno zdarzenie w danej chwili – założenie to pozwala jednoznacznie stwierdzić ile pracownik poświęcił czasu na wykonanie zlecenia.

- W przypadku zgłoszenia zawierającego niezarejestrowany znacznik, system zgłosi błąd – z powodu jednoznacznego powiązania zdarzenia z lokalizacją, niemożliwa jest obsługa zgłoszenia zawierającego identyfikator znacznika który nie został jeszcze wprowadzony do systemu.

- Istnieją 3 grupy uprawnień dostępu do systemu – system posiada rozbudowany moduł uwierzytelniający użytkowników. Stworzone zostały trzy kategorie dostępu. Każdy użytkownik posiadający login i hasło posiada również jeden z trzech poziomów uprawnień.

  Są to:

  - `ROLE_USER` – jest to dostęp pozwalający na zalogowanie się do panelu administracyjnego w trybie z ograniczonym wglądem do danych. Użytkownik ma możliwość korzystania z klienckiej aplikacji mobilnej i obsługi zdarzeń.
  - `ROLE_SUPER_USER` – rola ta została stworzona z przeznaczeniem dla pracownika wprowadzającego zgłoszenie do panelu administracyjnego, który nie ma prawa do nadawania uprawnień innym pracownikom. Może on też korzystać z mobilnej aplikacji administracyjnej.
  - `ROLE_ADMIN` – jest to kategoria posiadająca wszystkie poniższe uprawnienia. Administrator ma wgląd do wszystkich danych zgromadzonych w systemie oraz możliwość korzystania ze wszystkich funkcji.

	ROLE_USER	ROLE_SUPER_USER	ROLE_ADMIN
Odczyt danych	X	X	X
Zamykanie zdarzeń	X		X
Edycja danych		X	X
Dodawanie danych		X	X
Usuwanie danych			X
Nadawanie uprawnień			X
Nadawanie dostępu			X
Dostęp do aplikacji klienckiej	X	X	X
Dostęp do aplikacji administracyjnej		X	X
Dostęp do panelu	X	X	X

Tabela 3.1 Porównanie uprawnień użytkowników

- Nie może być sytuacji, w której z tym samym znacznikiem skojarzone są stworzone dwa zdarzenia o statusie utworzone/rozpoczęte – jest to spowodowane maksymalnym uproszczeniem interfejsu użytkownika aplikacji mobilnej, tak aby rejestracja rozpoczęcia lub zakończenia zdarzenia odbywała się w sposób jak najbardziej intuicyjny i nieangażujący.

- Każda lokalizacja jest jednoznacznie definiowana przez identyfikator znacznika – każde urządzenie kontrahenta posiada indywidualny znacznik naklejony na obudowę.

- Lokalizacje są trwale powiązane z kontrahentem – założenie to zwiększa przejrzystość prezentacji danych.

- Wszystkie dane lokalizacji są przechowywane w bazie danych – system nie dokonuje zapisu informacji charakterystycznych dla lokalizacji na znaczniku w celu uniknięcia rozproszenia i ujawnienia danych.

Schematy prezentujące logikę biznesową która realizuje te założenia, przedstawione zostały w rozdziale 5. Implementacja projektu.

3.3. SCENARIUSZ UŻYCIA

Aplikacja może znaleźć zastosowanie w firmach zajmujących się serwisowaniem urządzeń położonych w różnych zewnętrznych lokalizacjach należących do zewnętrznych kontrahentów.

Jako przykład przedstawiona zostanie sytuacja w której firma X zajmuje się serwisowaniem urządzeń wielofunkcyjnych – kopiarek, skanerów, drukarek.

Pracodawca, chcąc wdrożyć system, instaluje serwer usługi sieciowej i portalu informacyjnego i wprowadza dane pracowników do bazy. Do pracowników przypisuje urządzenia mobilne, definiując numery IMSI i IMEI. Każdy pracownik posiada również identyfikator i hasło do systemu oraz przydzieloną rolę

Pewna firma podpisuje z firmą X umowę na usługę serwisowania urządzeń biurowych na miejscu, w jej siedzibie.

W momencie podpisania umowy z kontrahentem, przy pomocy mobilnej aplikacji administracyjnej do systemu wprowadzane są podstawowe dane kontrahenta i urządzenia które ma być serwisowane, na obudowie którego umieszczana jest naklejka ze znacznikiem zbliżeniowym NFC i ewentualnym logiem firmy. W tym samym procesie unikatowy identyfikator znacznika wiązany jest w systemie z konkretnym urządzeniem.

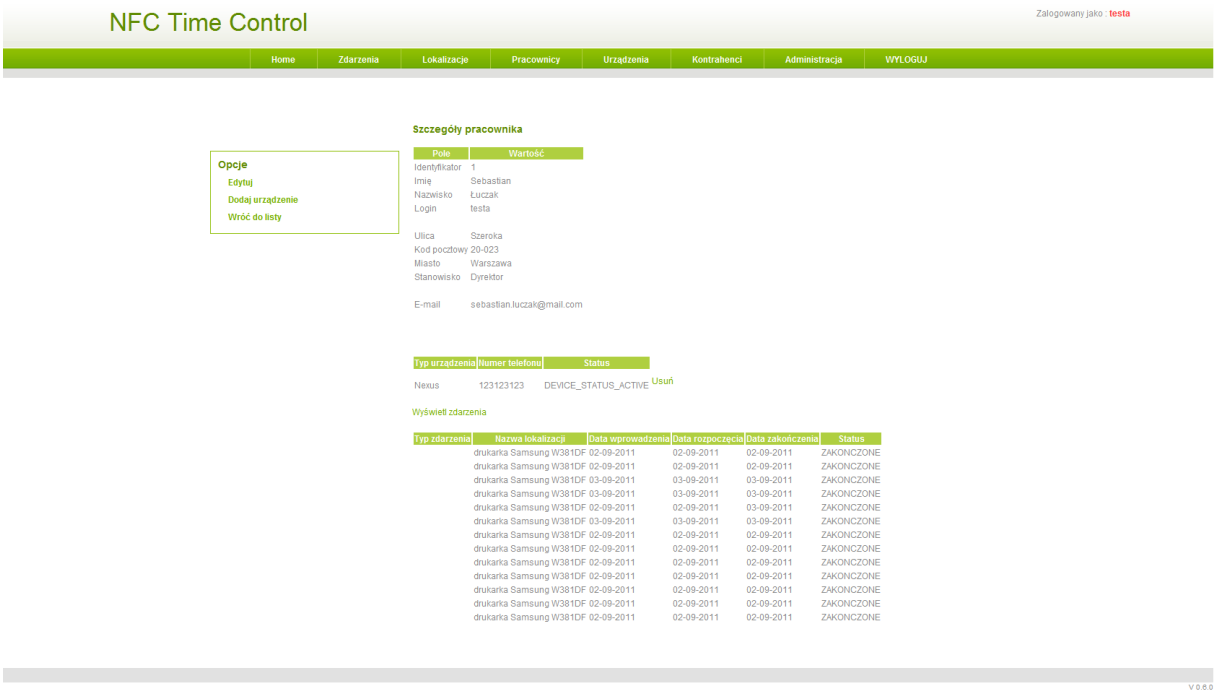
W sytuacji w której kontrahent zwraca się z potrzebą wykonania usługi serwisowej, osoba przyjmująca zlecenie (operator – osoba posiadająca uprawnienia `ROLE_SUPER_USER`), wprowadza dane do systemu i tworzy zdarzenie oczekujące na reakcję pracownika mobilnego. Pracownik mobilny otrzymuje powiadomienie w postaci wiadomości e-mail.

Na miejscu realizacji zlecenia pracownik mobilny zbliża telefon do znacznika naklejonego na obudowie obecnie serwisowanego urządzenia. Aplikacja zainstalowana na aparacie automatycznie odbiera dane ze znacznika i kontaktuje się z serwerem w celu zgłoszenia rozpoczęcia realizacji usługi. Serwer weryfikuje tożsamość użytkownika i rejestruje godzinę rozpoczęcia obsługi zdarzenia. Aplikacja na telefonie potwierdza zarejestrowanie danych na serwerze i wyświetla informacje dodatkowe wprowadzone wcześniej przez operatora.

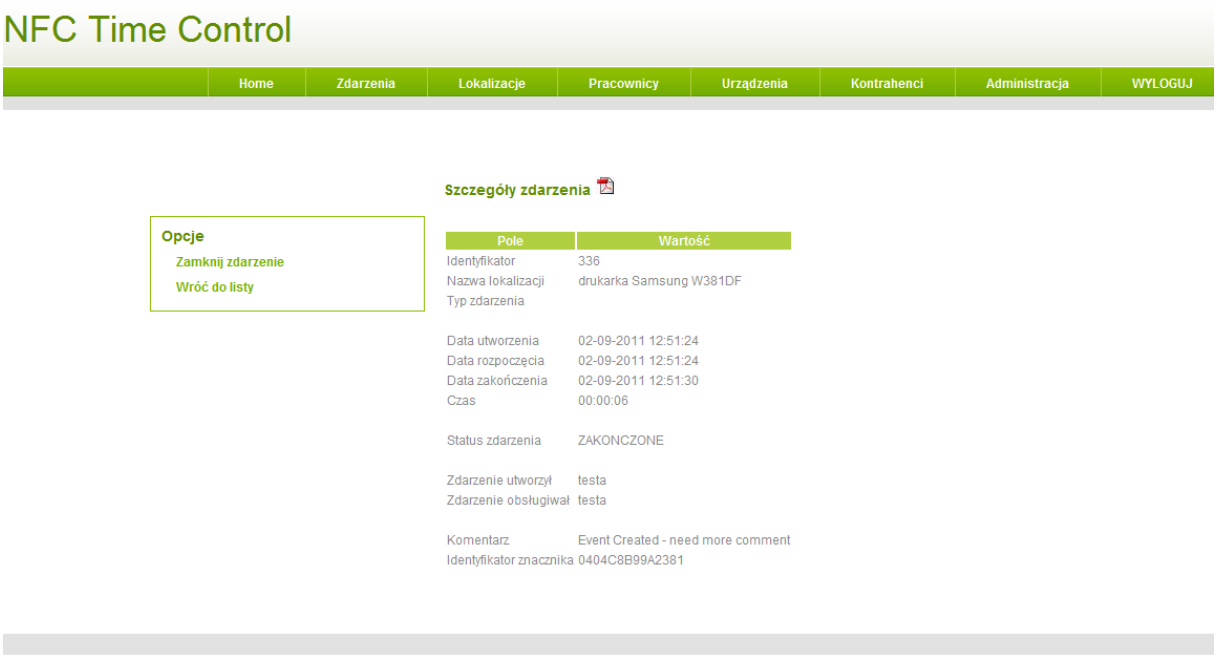
Po zakończeniu usługi pracownik mobilny ponownie przybliża telefon do znacznika i potwierdza chęć zakończenia realizacji zlecenia. Serwer potwierdza zakończenie zlecenia, odnotowuje moment końca pracy i zmienia stan zlecenia na zakończone.

Telefon sygnalizuje pomyślne zakończenie pracy i przełącza się w tryb oczekiwania na następne działanie.

Pracodawca, operator lub administrator może sprawdzić w panelu administracyjnym stany realizacji wszystkich zleceń oraz czas jaki został poświęcony na realizację konkretnej czynności przez danego pracownika.



Rys. 3.2 Wykaz zdarzeń zrealizowanych przez pracownika



Rys. 3.3 Szczegóły zrealizowanego zdarzenia

4. DOBÓR TECHNOLOGII

Początkowy etap projektu polegał na porównywaniu dostępnych technologii i doborze optymalnych dla realizacji systemu. Pierwsza faza obfitowała w liczne prototypy dzięki którym zostały wyeliminowane błędne założenia i elementy nadmiarowe, które nie spełniały oczekiwań.

4.1. NEAR FIELD COMMUNICATION

Użycie komunikacji zbliżeniowej zapewnić miało dużą łatwość obsługi systemu przez użytkowników końcowych, dzięki czemu mogliby oni się również łatwo przekonać do proponowanego rozwiązania.

Jako że najbardziej znanym standardem takiej komunikacji jest RFiD, początkowo rozważana była właśnie ta technologia. Na drodze obserwacji dostępnych rozwiązań, ustalono, że istnieją czytniki przenośne, ale są one drogie, relatywnie duże i nieporęczne. Biorąc pod uwagę fakt iż realizowane rozwiązanie miało pomagać pracodawcy i pracownikom, a nie utrudniać wykonywanie obowiązków, był to fakt nie do przyjęcia.

Ponieważ do standardowego wyposażenia docelowych użytkowników systemu, czyli pracowników mobilnych, należy telefon komórkowy, rozważona została możliwość integracji czytnika z urządzeniem tego typu. Działanie takie niesie za sobą korzyści w postaci uniknięcia wyposażania go w dodatkowe urządzenie i uzyskaniu dostępu do metod transmisji danych opartych o istniejącą infrastrukturę sieci komórkowych.

Near Field Communication stanowi grupę technologii krótkodystansowej radiowej komunikacji zbliżeniowej, stworzonych głównie z myślą o urządzeniach przenośnych takich jak telefony komórkowe i tablety [2]. Częstotliwość pracy wynosi 13.56 MHz, jest zatem taka sama jak jedna z częstotliwości używanych przez RFiD. Nie jest to przypadkowe, gdyż NFC jest w pełni kompatybilne z istniejącą już infrastrukturą RFiD opartą o tę częstotliwość.

Komunikacja w NFC oparta jest o zjawisko indukcji magnetycznej powstałej pomiędzy dwiema antenami. Zależnie od wariantu standardu, modulacja ASK lub BPSK tego pola pozwala na transmisję danych [3]. Dopuszczalne prędkości zawierają się w zakresie od 106 kbit/s do 424 kbis/s [3].

Transmisja zachodzić może w dwóch trybach, pasywnym i aktywnym.

W trybie pasywnym urządzenie inicjujące komunikację wytwarza pole elektromagnetyczne, z kolei drugie urządzenie moduluje powstałe pole i w ten sposób transmituje dane. Korzystając z tego samego pola, uzyskuje również moc niezbędną do zasilenia swoich układów.

Tryb aktywny oznacza, że oba urządzenia mają własne źródła zasilania i wytwarzają swoje pola na zmianę w celu cyklicznego nasłuchiwania i modulacji pola drugiego urządzenia.

Tryb pasywny opiera się głównie na wykorzystywaniu urządzenia mobilnego NFC, na przykład telefonu i pasywnego znacznika zgodnego z jednym z czterech typów wyspecyfikowanych przez standard. Dobór typu znacznika zależy ściśle od jego przeznaczenia, gdyż typy te różnią się między sobą oferowanymi prędkościami odczytu, pojemnością, systemami zabezpieczeń, obecnością unikalnego identyfikatora oraz kosztem jednostkowym [4].

W systemie kontroli czasu pracy technologia Near Field Communication użyta jest w trybie pasywnym do zbliżeniowej komunikacji telefonu ze znacznikami umieszczonymi w lokalizacjach realizacji zleceń przez pracowników.

4.2. ZNACZNIK NFC FORUM TYPE 2 - MIFARE ULTRALIGHT

Jedną z kluczowych zalet prezentowanego systemu jest jego niska cena wdrożeniowa i eksploatacyjna. Z tego powodu do oznaczania lokalizacji w których realizowane mogą być zlecenia zostały wybrane niezwykle tanie i proste konstrukcyjnie znaczniki NFC Forum Type 2 Mifare Ultralight w formie miękkiej nalepki, w której cały układ i antena wprasowane są w papier.

Znaczniki te cechują się pojemnością zaledwie 384 bitów dostępnych na dane użytkownika w trybie zapis/odczyt. Wyposażone są w system unikania kolizji oparty na analizie UID (ang. *Unique Identifier*) znacznika [5].

Mifare Ultralight oferuje prędkość transmisji 106 kbit/s i dostarcza możliwość jednokrotnego i nieodwracalnego zablokowania zapisu na znaczniku. Wadą tego rozwiązania jest niemożność ustalenia kodów dostępu do odczytu sektorów, przez co wszystko co jest zapisane na znaczniku, może być odczytane przez czytnik zgodny ze standardem ISO/IEC 14443 [6] [4].

Cechą która stała się krytyczna dla systemu okazał się fakt iż każdy znacznik posiada zapisany na 7 bajtach unikatowy numer seryjny, zwany dalej unikalnym identyfikatorem.

Ostatecznie na znaczniku zapisywana jest tylko nazwa systemu, a jedyną informacją odczytywaną ze znacznika jest jego identyfikator, który w bazie danych powiązany jest z kompletem potrzebnych informacji. Dokładny opis wspomnianych danych zawarty jest w rozdziale opisującym projekt bazy danych.

Rys. 4.1 Znacznik Mifare Ultralight w formie miękkie naklejki

4.3. ANDROID GINGERBREAD

Kolejnym istotnym czynnikiem doboru technologii w której zrealizowana miała być część pracy zwana mobilną, była popularność danego rozwiązania na rynku konsumenckim. Po zapoznaniu się z obecnym podziałem rynku okazało się, że 43,4% (Dane z Sierpnia 2011, Gartner [7]) obecnie sprzedawanych urządzeń typu smartphone, wyposażona jest w system operacyjny Android. Dodatkowo, po obserwacji dostępności terminali wyposażonych w system NFC, jasnym stało się, że również w tym wypadku system ten wiedzie prym, zarówno pod względem jakości dostarczonego API (ang. *Application Programming Interface*), uniwersalności, jak i liczby zapowiedzianych modeli wyposażonych w tę technologię.

Konsekwencją popularności Androida jest duży dostęp do różnorodnych opracowań na temat rozwoju i pielęgnacji aplikacji pisanych z przeznaczeniem na ten system oraz ogromne wsparcie ze strony użytkowników jak i samych twórców systemu. To zaowocowało dostępnością wielu bibliotek rozszerzających jego możliwości programistyczne.

Jako otwarta platforma Android daje programiście dostęp do obsługi praktycznie wszystkich podzespołów urządzenia i magazynów danych, a ścisłe wersjonowanie kodu daje gwarancję, że aplikacja napisana z przeznaczeniem na dane wydanie API, będzie działała tak samo na różnych modelach urządzeń przenośnych, spełniających wymogi zdefiniowane w pliku `AndroidManifest.xml` danej aplikacji [8].

W pierwszej fazie projektowania aplikacji trzeba było uwzględnić fakt iż system Android Gingerbread[1] 2.3 nie dostarczał interfejsu programistycznego pozwalającego na pełną obsługę NFC. Dopiero kolejne aktualizacje systemu operacyjnego dały dostęp do pełnej funkcjonalności. W wersji 2.3 niemożliwe było między innymi zapisywanie informacji na znaczniku NFC (tylko odczyt). Obsługiwana była mniejsza liczba dostępnych typów znaczników i nie istniała możliwość obsłużenia zbliżonego do urządzenia znacznika przez aplikację uruchomioną na pierwszym planie (system zawsze pytał użytkownika która aplikacja ma być wywołana dla danego znacznika) w sposób automatyczny [9].

Początkowy projekt aplikacji zakładał występowanie tych ograniczeń, szybko jednak okazało się, że w aktualizacji 2.3.3 (Luty 2011 [10]) wspomniane funkcje zostały dostarczone, a dostępne wcześniej API zmodyfikowane i w wielu przypadkach zauważalnie poprawione [11].

Po pierwszych prototypach wykorzystujących nowe możliwości, ustalono, że część zaproponowanych usprawnień może faktycznie pozytywnie wpłynąć na komfort użytkowania końcowej aplikacji, przyspieszając jej wywołanie i zapewniając reakcję tylko na znaczniki określonego typu. Chodzi tu o bibliotekę `android.nfc.tech`, w oparciu o którą system dokonuje wstępnego doboru puli aplikacji zdolnych do obsłużenia danego znacznika [9]. Dzięki tej funkcji udało się doprowadzić do sytuacji w której przy zbliżeniu do telefonu znacznika zarejestrowanego przez aplikację administracyjną, aplikacja kliencka uruchamia się automatycznie.

Metoda `enableForegroundDispatch()` odpowiedzialna za przekazywanie intencji zawierającej dane odczytanego znacznika do obecnie wywołanej aplikacji nie jest pozbawiona wad. W trakcie testów okazało się, że nie zawsze zachowuje się ona zgodnie z założeniem i system pyta użytkownika która aplikacja ma zostać wywołana, choć powinien przekazać dane do aplikacji pierwszego planu.

Korzystając z systemu Andoid Gingerbread monitorowane było również zużycie baterii przez moduł komunikacji NFC i procesy aplikacji. Wyniki przedstawione zostały w dziale opisującym testy funkcjonalne.

[1] Jest to piąta wersja systemu Android oznaczona numerem 2.3. Główne zmiany które wniosła to poprawki do interfejsu użytkownika, nowa klawiatura ekranowa i wsparcie dla Near Field Communication [11]

4.4. GOOGLE NEXUS S

Ponieważ pracownicy mobilni często wyposażeni są w zaawansowane urządzenia typu smartphone, jako docelową platformę dla aplikacji mobilnych wybrane zostało rozwiązanie tego typu. Dobór konkretnego modelu był zdeterminowany przez wybór systemu operacyjnego Android Gingerbread i wymóg posiadania modułu NFC. Badanie rynku wykazało, iż jedynym z najbardziej popularnych telefonów spełniającym te wymagania jest Google Nexus S. Po oficjalnej aktualizacji oprogramowania do wersji Android 2.3.4, telefon spełnił wszystkie złożone w nim oczekiwania.

4.5. HIBERNATE 3

Framework Hibernate w wersji 3 jest najbardziej popularnym rozwiązaniem programistycznym oferowanym na zasadach open source[2], które realizuje podejście mapowania obiektów na struktury danych składowanych w relacyjnych bazach danych (ang. ORM = *Object-to-Relational Mapping*). Hibernate ma za zadanie tłumaczyć dane przechowywane w postaci obiektów POJO (ang. *Plain Old Java Object*) do postaci rekordów w bazie danych, jak i odwrotnie. Umożliwia on zapisywanie oraz pobieranie obiektów Java bez konieczności pisania zapytań w języku SQL (ang. *Structured Query Language*).

Dzięki niewielkiemu nakładowi pracy potrzebnemu na konfigurację, możliwą zarówno za pośrednictwem plików xml jak i adnotacji zamieszczonych bezpośrednio w kodzie aplikacji, programista skupia się na logice wytwarzanego oprogramowania, podczas gdy Hibernate dba o zarządzanie sesją, transakcjami oraz o poprawne przechowywaniu danych z obiektów w encjach bazy danych.

Ze względu na różnicę pomiędzy bazami danych, framework wprowadza mechanizm dialektów, który je ukrywa przed programistą za swoim interfejsem.

Hibernate udostępnia programistom bardzo szerokie API ułatwiające tworzenie zapytań, konstruowanie filtrów a także stronicowanie wyników pobieranych z bazy danych. Dzięki takim udogodnieniom jak HQL (ang. *Hibernate Query Language*), QBE API (ang. *Query By Example*) czy też Criteria API, operacje na rekordach bazy danych stają się o wiele łatwiejsze w porównaniu do wykorzystywania standardowego połączenia z bazą danych JDBC (ang. *Java DataBase Connectivity*).

[2] Oprogramowanie rozpowszechniane na zasadach tej licencji jest darmowe, posiada ogólnodostępny kod źródłowy oraz może być modyfikowane.

W stworzonym systemie, Hibernate pełni funkcję głównego ogniwa łączącego warstwy logiki biznesowej z warstwą modelu danych. Wykorzystywany jest zarówno w aplikacji usługi sieciowej jak i aplikacji panelu administracyjnego.

4.6. SPRING FRAMEWORK 3

Kolejnym frameworkiem wykorzystywanym w pracy jest Spring w wersji 3.0.1. Stwarza on środowisko, które łączy ze sobą w harmonijną całość liczne interfejsy API.

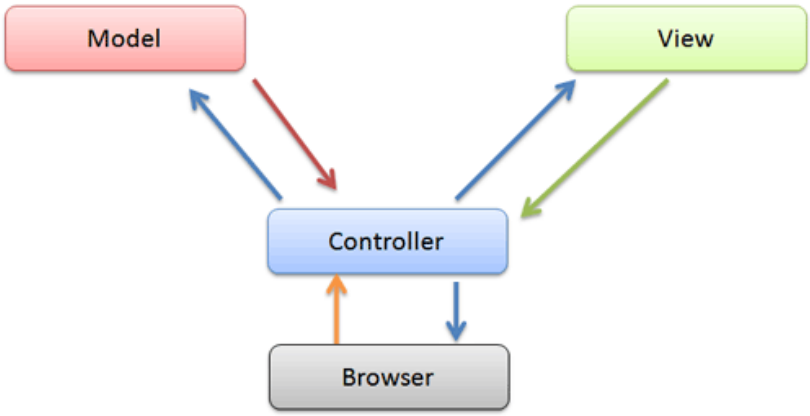
Dzięki mechanizmowi wstrzykiwania zależności (ang. *dependency injection*), Spring udostępnia cały szereg klas opakowujących popularne biblioteki Javy, przez co ich użycie staje się znaczenie prostsze.

Dodatkowo oferuje wsparcie dla wszystkich elementów potrzebnych do stworzenia zarówno aplikacji desktopowej, jak i aplikacji internetowej. Implementuje sprawdzone wzorce projektowe, a dzięki doskonale stworzonej dokumentacji ułatwia i promuje dobre praktyki programistyczne [12].

W jego skład wchodzą również bardzo rozwinięte klasy wspierające integrację z innymi szkieletami aplikacyjnym takimi jak np. Hibernate czy też Log4j.

Spring, składa się z wielu modułów, które mogą być stosowane niezależnie. Są nimi między innymi pakiety Spring MVC oraz Spring Security, które wykorzystywane są w systemie.

Spring MVC (ang. Model – *View* – *Controller*) jest jedną z implementacji wzorca projektowego Model – Widok – Kontroler. Zakłada on rozdzielenie warstwy modelu danych oraz logiki biznesowej od warstwy widoku, najczęściej stron JSP (ang. *Java Server Pages*) odpowiedzialnych za wyświetlanie danych. Według wzorca warstwa biznesowa, na rysunku zawarta w module kontrolera, powinna odwoływać się do danych wyłącznie za pośrednictwem warstwy modelu danych. Dzięki takiemu rozwiązaniu warstwa biznesowa nie musi posiadać wiedzy o tym czy do składowania obiektów stosowana jest baza danych czy też system plików [13]. Przepływem pomiędzy tymi warstwami zarządza kontroler, który także otrzymuje dane wejściowe od użytkowników i zarządza generowaniem odpowiednich widoków.



Rys. 4.2 Ilustracja wzorca MVC

Spring Security to inny pakiet projektu Spring używany w systemie. Jego zadaniem jest zapewnienie bezpieczeństwa panelu administracyjnego. Dzięki udogodnieniom oferowanym przez ten pakiet, w łatwy sposób można kontrolować dostęp użytkowników do konkretnych funkcjonalności portalu. Moduł ten jest odpowiedzialny za uwierzytelnianie użytkowników a więc potwierdzenie ich tożsamości oraz za autoryzację, czyli udzielenie bądź też pozbawienie dostępu do określonych części systemu.

Wspiera on wiele metod uwierzytelnienia, takich jak na przykład LDAP czy też protokół Kerberos. W prezentowanym rozwiązaniu uwierzytelnianie użytkownika opiera się na sprawdzeniu jego danych z informacjami zapisanymi w bazie danych.

4.7. ORACLE 10G EXPRESS EDITION

W realizowanym projekcie wykorzystany został system zarządzania bazą danych Oracle w wersji 10g Express Edition, stworzony przez firmę Oracle Corporation. Edycja XE to darmowa, ograniczona funkcjonalnie wersja serwera bazodanowego Oracle 10g. Nie jest ona w stanie obsłużyć więcej niż jednego procesora oraz więcej niż 4GB pamięci RAM.

Pomimo swoich ograniczeń, serwer zachowuje wydajność oraz niezawodność. Dzięki rozbudowanym możliwościom administracji oraz programom pokrewnym, takim jak Oracle SQL Developer czy też Oracle Database Modeler, możliwą jest łatwa budowa schematu bazy danych, a także jego późniejsza implementacja.

Baza danych firmy Oracle wybrana została ze względu na chęć stworzenia systemu opartego na profesjonalnych komponentach składowych. Jednym z jej atutów jest przenośność na różne platformy sprzętowe i systemowe. Według danych za rok 2011 Oracle Corporation posiada niemal 50% rynku baz danych na świecie [14]. Dzięki swojej

popularności nie ma żadnych problemów w integracji bazy danych z frameworkiem Hibernate.

4.8. JAVA 2 ENTERPRISE EDITION

Java jest obiektowym językiem programowania udostępnionym przez firmę Sun Microsystems w 1995 roku. Dzięki swojej prostocie, a jednocześnie zaawansowanym możliwościom, szybko zyskała miano jednego z najpopularniejszych języków programowania na świecie.

Największą zaletą języka Java, jest jego niezależność od platformy sprzętowej. Programy stworzone w tym języku, można bez problemu uruchomić na wszystkich systemach operacyjnych, które posiadają wirtualną maszynę Java JVM (ang. *Java Virtual Machine*).

Java 2 EE jest szeroko rozpowszechnioną oraz używaną platformą programistyczną opracowaną dla języka Java. Definiuje ona standardy oraz wzorce projektowe dla tworzenia oprogramowania opartego o wielowarstwową architekturę komponentową uruchamianego na serwerach aplikacyjnych. Jest to rozwinięcie koncepcji „napisz raz, a uruchomisz wszędzie", obowiązującej w programowaniu aplikacji WWW [13]. W jej skład wchodzą serwery, komponenty EJB (ang. *Enterprise JavaBeans*), usługi Web Service, pliki JSP, XML (ang. *eXtesible Markup Language*) oraz inne elementy ściśle związane z wytwarzaniem aplikacji WWW takie jak HTML (ang. *HyperText Markup Language*) i CSS (ang. *Cascading Style Sheets*).

Standard J2EE obejmuje liczne interfejsy wspomagające programistę np. przy zapewnianiu bezpieczeństwa, definiowaniu interfejsu użytkownika czy też wysyłaniu poczty elektronicznej.

4.9. APACHE TOMCAT

Nierozłącznym elementem aplikacji WWW jest serwer. W przedstawianym projekcie wykorzystywana jest rozpowszechniana na zasadzie open source implementacja Apache Tomcat 6.0.32. Jest to kontener aplikacji webowych (ang. *Web container*) będący serwerem, który umożliwia uruchamianie aplikacji internetowych wykonanych w technologiach Java servlets lub też JSP. Jest on w pełni kompatybilny z wymaganiami aplikacji opartych na J2EE.

4.10. ARCHITEKTURA REST

REST (ang. *Representational State Transfer*) jest wzorcem architektonicznym opartym na standardzie aplikacji internetowych tworzonych według modelu klient-serwer, komunikujących się poprzez protokół HTTP. Wprowadza on szereg usprawnień i udogodnień mających na celu poprawienie wydajności, skalowalności oraz edycyjności usług internetowych (ang. *Web Services*).

Podstawowym założeniem wspomnianej architektury jest traktowanie danych oraz funkcji jako zasobów łatwo dostępnych poprzez identyfikatory URI (ang. *Uniform Resources Identifiers*), będących najczęściej hiperłączami w sieci. Serwer albo usługa WWW reaguje na żądania i dzięki mechanizmowi negocjacji zawartości (ang. *content negotiation*) dobiera możliwie najlepszą reprezentację danych spełniającą warunki żądania. Zaletą takiego rozwiązania jest możliwość przesyłanie zasobów posiadających wiele różnych reprezentacji np. plain-text, xml, json. Innym kluczowym pomysłem wykorzystywanym w architekturze REST jest pojęcie jednorodnego interfejsu (ang. *uniform interface*), czyli standardowego zbioru metod wykorzystywanych do określonych czynności. W protokole HTTP najczęściej stosowane są metody PUT, GET, POST, DELETE odpowiadające operacjom CRUD – (Create, Retrieve, Update, Delete) znanym z baz danych [13].

Udogodnienia te sprawiają, że aplikacje oparte na wzorcu REST stają się proste, lekkie, oraz bardzo wydajne.

Wzorzec REST jest alternatywą dla dobrze znanego standardu SOAP (ang. *Simple Object Access Protocol*), oferującego cały wachlarz zaawansowanych możliwości, będącego neutralnym dla protokołów warstwy transportowej. Niestety, główną wadą standardu SOAP jest potrzeba złożonej konfiguracji oraz opisu.

Użyta architektura posiada wiele implementacji, jedną z nich jest projekt Jersey, wykorzystywany w poniższej pracy do stworzenia usługi internetowej komunikującej się z telefonem komórkowym.

Jersey jest wolno dostępną implementacją JAX-RS (ang. *Java API for RESTful Web Services*), udostępniającą mechanizm adnotacji, który to znacznie ułatwia programistom tworzenie aplikacji.

Dodatkowym atutem przy wyborze tego wzorca jest fakt, iż współcześnie jest to niezwykle popularna architektura przy programowaniu aplikacji na system Android.

4.11. JSON - JAVASCRIPT OBJECT NOTATION

Przy wyborze formatu danych transmitowanych między aplikacjami mobilnymi i usługą sieciową główny nacisk położony został na prostocie, niezależności od platformy i łatwości wyszukiwania błędów. Z tego powodu wykorzystany został format JSON. Jest to prosty format danych służący do wymiany informacji, możliwy do odczytania przez człowieka. Początkowo został stworzony dla języka JavaScript lecz szybko stał się niezależnym formatem tekstowym posiadającym implementacje w wielu językach programowania, między innymi Java. Jego właściwości sprawiają, że jest idealnym narzędziem do transmisji uporządkowanych danych poprzez sieć [15].

JSON jest alternatywą dla powszechnie wykorzystywanego w Internecie języka XML, który wymaga znacznie większych nakładów programistycznych – dodatkowe klasy i obiekty – przy przetwarzaniu danych. Tabela 4.1 porównuje oba te formaty.

W realizowanym systemie, po stronie usługi internetowej obsługą wiadomości w formacie JSON zajmuje się framework Jersey, z kolei po stronie aplikacji mobilnych biblioteka GSON. Stworzona na potrzeby własne programistów Google, biblioteka ta została ostatecznie opublikowana do użytku ogólnego i tak jak sam format JSON, jest niezwykle prosta w użytkowaniu [16].

Kryterium	JSON	XML
Przejrzystość dla człowieka	Porównywalna	
Łatwość tworzenia (strona serwera)	Porównywalna	
Łatwość przetwarzania (strona klienta)	Natychmiastowa i bezproblemowa (niezbędna jedna linia kodu)	Wymaga poruszania się po drzewie DOM, podatna na występowanie błędów
Rozszerzalność typów zawartości	Bezpieczne i proste, wystarczy dodać warunek na występowanie nowej wartości	Wymaga zmian w kodzie, inaczej aplikacja przestanie funkcjonować poprawnie
Wyszukiwanie błędów	Ręczne po stronie klienta Ręcznie po stronie serwera	Ręczne po stronie klienta Automatycznie po stronie serwera (walidacja)
Bezpieczeństwo	Porównywalne	

Tabela 4.1 Porównanie formatów JSON i XML

Poniżej widnieje przykładowy format danych, przesyłany w pomiędzy telefonem komórkowym a usługą sieciową.

```
//Instancja klasy CResponse zapisana w notacji JSON:
{
  „STATUS” : „EVENT_STATUS_CREATED”,
  „MESSAGE” : „przykładowa wiadomość”,
  „TOKEN”: „2cabeae9a6c1febc74f2f8d9af6391104952445f”
}
```

4.12. SYSTEM KONTROLI WERSJI SUBVERSION 1.6

Projekt rozwijany zespołowo wymaga dodatkowych narzędzi ułatwiających współpracę, w tym możliwości kontroli zmian zachodzących w kodzie i łączenia pracy wykonanej przez różnych członków zespołu.

W procesie rozwoju projektu wykorzystane zostało sprawdzone rozwiązanie w postaci repozytorium Subversion 1.6 udostępnione na zasadach licencji GPL[3] (ang. *General Public Licence*). Dzięki temu możliwe jest łatwe współdzielenie kodu aplikacji i nanoszenie zmian przez dwie osoby w tym samym momencie, jednocześnie zachowując porządek i możliwość wycofania modyfikacji w sytuacji gdy nie przyniosły one oczekiwanych efektów.
Subversion to rozwiązanie oparte o architekturę klient-serwer, które do bezproblemowej i szybkiej pracy wymaga hostingu o dużej gwarantowanej przepustowości [17]. Z tego powodu wykorzystana została usługa *code.google.com*, która prócz innych rozwiązań ułatwiających prowadzenie projektu, udostępnia również repozytorium Subversion w wersji 1.6.

4.13. ECLIPSE SDK 3.6.2 + JAVA EE

Środowisko programistyczne Eclipse powstało w wyniku narastającego zapotrzebowania programistów na narzędzia wspomagające ich pracę przy tworzeniu coraz to bardziej zaawansowanego oprogramowania. Z biegiem czasu Eclipse stał się najpopularniejszym narzędziem pracy developerów na całym świecie.

Projekt Eclipse stworzony prze firmę IBM w 2001 roku, w 2004 został udostępniony na zasadzie open source. Obecnie jest to platforma wpierająca najpopularniejsze języki programowania. Dzięki możliwościom rozszerzenia jej funkcjonalności, za pomocą licznych wtyczek (ang. *Plug-ins*) możliwa jest łatwa współpraca z wieloma szkieletami programistycznymi.

Wykorzystanie Eclipse IDE (ang. *Integrated Development Environment*) udostępniło również możliwość wykorzystania zawartej w nim platformy Web Tools Platform, która to projekt promuje zasadę neutralności wobec producentów i pozwala na realizowanie aplikacji dla wielu różnych serwerów aplikacyjnych. Dzięki takiemu podejściu możliwe jest dogłębne przetestowanie aplikacji na różnych środowiskach oraz zgodność ze standardem i przenośność pomiędzy nimi.

[3] Licencja GNU GPL wymaga od programu spełnienia czterech podstawowych zasad wolności. Są to: wolność uruchamiania programu w dowolnym celu; wolność analizowania jak program działa i dostosowywania go do swoich potrzeb; wolność rozpowszechniania niezmodyfikowanej kopii programu; wolność udoskonalania programu i publicznego rozpowszechniania własnych ulepszeń

4.14. ECLIPSE SDK 3.6.2 + ANDROID SDK

Ponieważ programowanie na system Android wymaga jednoczesnej edycji wielu plików źródłowych i zasobów zapisanych w formacie .xml, a dodatkowo używania automatycznie generowanej klasy R (Resource), korzystanie z samego SDK (ang. *Source Development Kit*) dostarczonego przez Google jest bardzo uciążliwe [18]. W celu usprawnienia pracy nad aplikacjami mobilnymi wykorzystane zostało zintegrowane środowisko programistyczne Eclipse SDK 3.6.2 połączone ze wspomnianym zestawem narzędzi Android SDK.

Dzięki temu połączeniu możliwy jest dostęp do złożonej aplikacji typu debugger, wgląd w wydruki konsoli systemu Android poprzez narzędzie LogCat, a praca z kodem źródłowym została znacznie przyspieszona dzięki narzędziom podpowiadania składni [9].

Dodatkowa integracja z klientem Subversion pozwoliła przy pomocy jednego narzędzia realizować wszystkie potrzebne działania.

5. IMPLEMENTACJA PROJEKTU

5.1. BAZA DANYCH

5.1.1. Cel powstania

Baza danych jest podstawowym elementem opracowanego systemu. Jej zadaniem jest przechowywanie danych o zdarzeniach, użytkownikach, urządzeniach, lokalizacjach oraz kontrahentach, tak aby móc zapewnić poprawne działanie panelu administracyjnego oraz usługi sieciowej.

5.1.2. Szczegóły implementacji

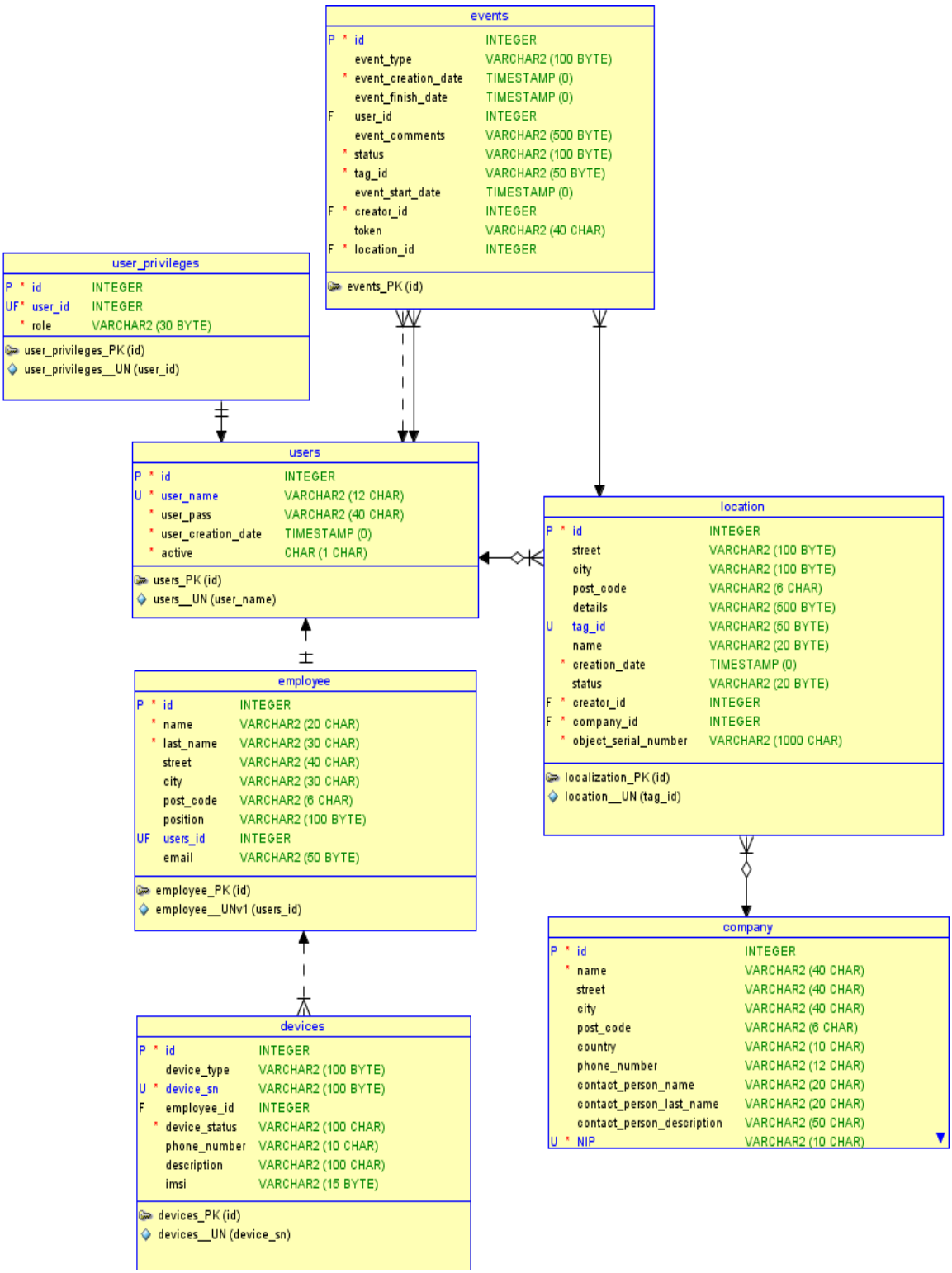
Baza została stworzona w języku SQL na serwerze Oracle 10g Express Edition w oparciu o relacyjny model danych. Twórcą tego modelu jest Edgar Frank Codd, który w swojej publikacji A Relational Model of Data for Large Shared Data Banks [19], opisał po raz pierwszy zasady grupowania danych oraz panujących pomiędzy nimi zależności.

Model relacyjny zakłada grupowanie danych w relacje, które są reprezentowane za pomocą tabel. Są one zbiorem rekordów o ustalonej i identycznej strukturze wewnętrznej. Każda z nich posiada nazwę, nagłówek oraz zbiór atrybutów, zawartością relacji są z kolei krotki. Krotka jest jednoznacznie identyfikowana przez klucz główny (ang. *Primary Key*) będący unikatową wartością.

Ważnym elementem przedstawianego modelu jest klucz obcy (ang. *Foreign key*), będącym zbiorem wartości jednej tabeli wskazującym na wartości z innej tabeli. Jego podstawowym zadaniem jest pokazanie zależności między danymi składowanymi w różnych tabelach. Klucze obce, w modelu relacyjnym, służą również do zapewnienia spójności danych w bazie, muszą one spełniać wymóg, który mówi, że w tabeli wskazywanej musi istnieć wartość klucza wskazującego.

Językiem służącym do tworzenia zapytań oraz struktury bazy danych jest SQL.

Dzięki zaawansowanym możliwościom bazy danych Oracle, stworzone zostały mechanizmy zabezpieczające bazę przed wprowadzeniem błędnych danych, danych niezgodnych z przyjętymi założeniami oraz powtarzających się rekordów.



Rys. 5.1 Schemat stworzonej bazy danych

5.1.3. Opis tabel

Poniżej przedstawiono skrótowy opis poszczególnych tabel z powyższego diagramu. Dokładny opis każdej z tabeli i jej pól został zamieszczony w dodatku A.

Tabela *users* gromadzi dane użytkowników mających dostęp do systemu. Zawiera ona login, skrót hasła utworzony przy pomocy algorytmu *SHA-1* (ang. *Secure Hash Algorithm*), datę zarejestrowania oraz status użytkownika.

Tabela *user_privileges*, przechowuje identyfikator użytkownika, będący kluczem obcym z tabeli *users* oraz rolę przypisaną do użytkownika. Relacja pomiędzy tabelą *users* a *user_privileges* jest relacją 1 do 1, a więc tylko jedna rola może zostać przypisana danemu użytkownikowi. W tabeli mogą być przechowywane tylko określone role. Wykaz roli wraz z ich opisem został już wcześniej przedstawiony w rozdziale o koncepcji systemu.

Tabela *employee* jest główną tabelą przechowującą informacje szczegółowe na temat pracowników zarejestrowanych w systemie. Zawiera identyfikator użytkownika, będący kluczem obcym pozwalający na określenie nazwy użytkownika i hasła danego pracownika. Relacja pomiędzy tabelą *users* a *employee*, jest relacją 1 do 1, przez co niemożliwa jest sytuacja, aby użytkownik miał przypisany więcej niż jeden login do systemu.

Tabela *devices,* ma za zadanie magazynowanie danych urządzeń zarejestrowanych w systemie. Posiada klucz obcy do tabeli *employee* w relacji wiele do jednego, dzięki któremu możliwy jest mechanizm przypisania urządzenia do danego pracownika. Pracownik może posiadać wiele urządzeń ale już konkretne urządzenie, może zostać przypisane tylko jednemu pracownikowi. Każde urządzenie zarejestrowane w systemie musi posiadać status zgodny z przyjętą konwencją.

Tabela *events* zawiera dane opisujące zdarzenia rejestrowane przez usługę sieciową lub panel administracyjny. Każde zdarzenie obowiązkowo musi posiadać status, datę utworzenia, identyfikator znacznika przypisany do danej lokalizacji oraz identyfikator użytkownika, który zarejestrował dane zdarzenie. Tabela events, posiada trzy klucze obce, dwa do tabeli *users* – mające za zadanie przechowywać identyfikatory użytkownika, który utworzył zdarzenie oraz użytkownika który danym zdarzeniem się zajmuję. Trzecim kluczem obcym jest identyfikator lokalizacji na którą zarejestrowane jest zdarzenie.

Tabela *locations* jest miejscem, w którym przechowywane są dane jednoznacznie wiążące unikalny identyfikator znacznika, nazwę lokalizacji oraz numer seryjny urządzenia. Każda lokalizacja jest powiązana w relacji jeden do wielu z tabelą *company*. Oznacza to, że każda lokalizacja jest jednoznacznie powiązana z firmą, jednocześnie na daną firmę może zostać zarejestrowane wiele lokalizacji.

Tabela *company* zawiera informacje szczegółowe opisujące kontrahentów zarejestrowanych w panelu administracyjnym. Każdy kontrahent jest jednoznacznie określony za pomocą numeru NIP.

5.1.4. Integralność danych

W celu zapewnienia spójności i integralności danych przechowywanych w bazie został zaimplementowany szereg ograniczeń, tak aby niemożliwe było stworzenie rekordów z błędnymi wpisami.

Za pomocą klauzuli *check*, założonej na tabelach *events*, *devices* oraz *locations* dokonuje się sprawdzenia czy wpisywane statusy obiektu są zgodne z założonymi :

STATUS	OPIS
LOCATION_STATUS_CREATED	lokalizacja utworzona
LOCATION_STATUS_ACTIVE	lokalizacja aktywna
LOCATION_STATUS_INACTIVE	lokalizacja nieaktywna
EVENT_STATUS_CREATED	zdarzenie utworzone
EVENT_STATUS_STARTED	zdarzenie rozpoczęte
EVENT_STATUS_SUSPENDED	zdarzenie zawieszone
EVENT_STATUS_FINISHED	zdarzenie zakończone
EVENT_STATUS_CLOSED	zdarzenie zamknięte
DEVICE_STATUS_CREATED	urządzenie utworzone
DEVICE_STATUS_ACTIVE	urządzenie aktywne
DEVICE_STATUS_INACTIVE	urządzenie nieaktywne

Tabela 5.1 Lista używanych statusów obiektów

Dodatkowy mechanizm obejmujący tabelę *events*, nie pozwala na zamieszczenie w niej dwóch wpisów z tym samym `TAG_ID` będących w statusie `EVENT_STATUS_STARTED` lub `EVENT_STATUS_CREATED`.

Poniższe tabele pokazują jakie sytuacje nie mogą mieć miejsca w bazie danych:

TAG_ID	STATUS
Abc	EVENT_STATUS_CREATED
Abc	EVENT_STATUS_STARTED
Abc	EVENT_STATUS_FINISHED

Tabela 5.2 Przedstawia sytuację poprawną

TAG_ID	STATUS
Abc	EVENT_STATUS_CREATED
Abc	EVENT_STATUS_CREATED
Abc	EVENT_STATUS_FINISHED

Tabela 5.3 Przedstawia sytuację niepoprawną

Poniżej znajduje się kod źródłowy SQL tworzący opisany przed chwilą mechanizm

```
create unique index event_created
    on events(
          (CASE WHEN status = 'EVENT_STATUS_CREATED'THEN tag_id
                ELSE NULL
            END)
            );

create unique index event_started
    on events(
          (CASE WHEN status = 'EVENT_STATUS_STARTED' THEN tag_id
                ELSE NULL
            END)
            );
```

Kod źródłowy 5.1 Zapewnienie poprawności magazynowanych zdarzeń

5.1.5. Wyzwalacze i sekwencje

W celu zapewnienia unikalności kluczy głównych dla każdej tabeli został stworzony wyzwalacz (ang. *Trigger*) oraz sekwencję które mają na celu zadbanie o generowanie pól ID.

Poniżej znajduje się przykładowy kod sekwencji oraz wyzwalacza :

```
create sequence events_id_seq start with 1 increment by 1;

create or replace trigger events_insert
before insert on events
for each row
begin
   if :new.id is null then
    select events_id_seq.nextval into :new.id from dual;
end if;
end;
```

Kod źródłowy 5.2 Przykład wyzwalacza oraz sekwencji tabeli *events*

5.2. OPIS USŁUGI SIECIOWEJ – WEB SERVICE

5.2.1. Cel powstania

Usługa sieciowa jest elementem zapewniającym komunikację pomiędzy zainstalowanymi na telefonach komórkowych aplikacjami mobilnymi a bazą danych i pozostałymi komponentami systemu.

5.2.2. Szczegóły implementacyjne

Stworzona usługa sieciowa jest samodzielną aplikacją zainstalowaną na serwerze, której zadaniem jest nasłuchiwanie i przetwarzanie zgłoszeń napływających od aplikacji klienckich zainstalowanych na telefonach pracowników mobilnych. Można wyróżnić jej cztery główne funkcje pozwalające kolejno na:

- uwierzytelnienie pracownika i umożliwienie mu dalszej pracy z aplikacją mobilną
- sprawdzenie przez aplikację kliencką czy w bazie danych istnieją zdarzenia będące w statusie „rozpoczęte”
- rejestrację nowego zdarzenia, lub też zamknięcie już istniejącego
- wprowadzenie do systemu nowej lokalizacji wraz z nowym identyfikatorem znacznika

Zaimplementowane rozwiązanie składa się z szeregu klas zgromadzonych w 5 modułach zapewniających poprawne działanie powyższych funkcji.

- Moduł DAO
- Moduł Entity
- Moduł WebService
- Moduł WebService.Request
- Moduł WebService.Response

Obiekty DAO (ang. *Data Access Object*) odpowiadają za wykorzystanie szkieletu aplikacji Hibernate 3, którego zadaniem jest poprawne mapowanie obiektów biznesowych na encje bazy danych.

Wraz z obiektami biznesowymi w module Entity przechowywane są pliki odwzorowań hibernate pozwalające na odpowiednie odtworzenie tabel i ich kolumn na obiekty klas entity.

Poniżej został zamieszczony przykładowy plik mapowań kolumn na obiekty klasy CUsers. Przedstawiono na nim w jaki sposób nazwy zmiennych, oraz ich typy są mapowane na nazwy kolumn z encji bazy danych. Kolejnym istotnym elementem tego pliku jest

zaznaczenie w jaki sposób mają być generowane klucze główne – unikatowe w skali tabeli identyfikatory. W tym przypadku za generowanie pól *ID* odpowiedzialna jest sekwencja *users_id_seq.*

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 2011-06-22 19:16:38 by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>

<class name="db.entity.CUsers" table="USERS">
   <id name="Id" type="long" column="id"   >
            <generator class="sequence">
            <param name="sequence">users_id_seq</param>
            </generator>
   </id>

<property name="userName" type="java.lang.String" column="USER_NAME"/>
<property name="userPassword" type="java.lang.String" column="USER_PASS"/>
<property name="creationDate" type="java.sql.Timestamp"
column="USER_CREATION_DATE" />
<property generated="never" lazy="false" name="isActive"
type="java.lang.Character" column="ACTIVE" sql-type="CHAR"/>
</class>
</hibernate-mapping>
```

Kod źródłowy 5.3 Plik mapowań hibernate klasy CUsers

Moduł webservice zawiera klasy będące implementacją usługi sieciowej w rozumieniu architektury REST. Każda z nich świadczy określone funkcje dla aplikacji mobilnej.

Dwa ostatnie moduły zawierające klasy `Request` oraz `Response`, przechowują obiekty będące odwzorowaniami zapytań i odpowiedzi przesyłanymi w formacie JSON pomiędzy komunikującymi się stronami.

Dodatkowo aplikacja zawiera plik konfigurujący `hibernate.cfg.xml`, który określa informacje potrzebne do połączenia się z bazą danych, a także pulę połączeń czy też dodatkowe funkcję hibernate takie jak możliwość logowania zapytań wysyłanych do bazy danych do pliku bądź też sposoby zwiększania szybkości realizacji zapytań przesyłanych do serwera bazodanowego.

W dalszej części tego rozdziału zostanie opisana każda z wyżej wymienionych funkcji a także będą zaprezentowane przypadki wykorzystania ich w aplikacji. Zostaną także przedstawione algorytmy obsługujące nadchodzące zgłoszenia.

5.2.3. Mechanizm uwierzytelnienia

Mechanizm uwierzytelnienia opiera się na wysłaniu zapytania przez aplikację mobilną na odpowiedni adres usługi sieciowej. Zapytanie ma na celu ustalenie czy dany użytkownik jest zarejestrowany w systemie, oraz czy urządzenie, z którego wysyła zapytanie jest przypisane właśnie do niego.

Aplikacja mobilna przesyła zapytanie na adres `/nfcTimeControl/service/auth` które zawiera login, hasło, numer IMEI telefonu komórkowego oraz numer IMSI będący identyfikatorem karty SIM włożonej do aparatu.

Przesłane dane pozwalają jednoznaczne określić uprawnienia użytkownika, jego dane, to czy nie jest zablokowany przez administratora systemu oraz to, czy urządzenie którym posługuje się pracownik jest zarejestrowane na jego nazwisko.

Implementując ten mechanizm wykraczający ponad sprawdzenie jedynie poprawności nazwy użytkownika oraz hasła, chcemy mieć pewność, że dane zdarzenie jest obsługiwane konkretnie przez pracownika będącego właścicielem telefonu. Numery IMEI oraz IMSI są sprawdzane ze względu na chęć zabezpieczenia się przed możliwością zamiany aparatów, czy też karty SIM pomiędzy pracownikami mobilnymi.

5.2.4. Mechanizm sprawdzania statusu zdarzenia

Zaimplementowany został mechanizm pozwalający na sprawdzenie czy w bazie danych istnieją zdarzenia otwarte i przypisane już dla zgłaszającego się użytkownika mobilnego. Pozwala to na zabezpieczenie aplikacji przed błędnym działaniem i potraktowaniu zgłoszenia będącego de facto zamknięciem już istniejącego jako nowe.

Przyczyną jego powstania jest fakt iż dane aplikacji mobilnej mogą zostać wyczyszczone przez pracownika z poziomu systemu operacyjnego telefonu lub podczas zmiany urządzenia, co mogłoby spowodować nieprawidłowe działanie systemu. Dzięki wyżej wymienionemu mechanizmowi, telefon automatycznie uzupełnia informacje potrzebne mu do dalszego funkcjonowania.

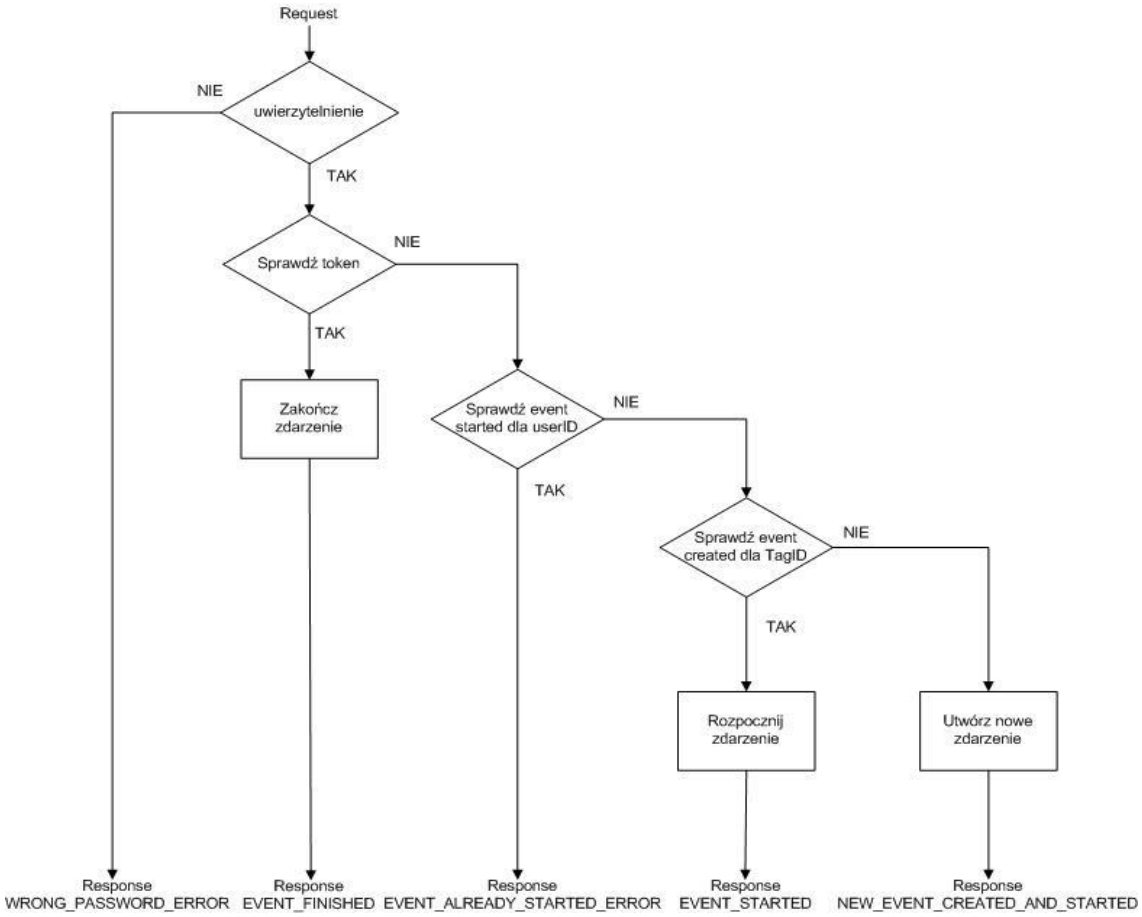
Każde zapytanie o status, będące reprezentacją klasy `CRequestStatus`, zawiera dane potrzebne do ponownego uwierzytelnienia użytkownika i jednocześnie pozwalające na odpytanie bazy danych o zdarzenia w statusie `EVENT_STATUS_STARTED` przypisane do go danego pracownika.

5.2.5. Rejestracja nowego zdarzenia

Mechanizm rejestracji nowego zdarzenia oraz odnotowania zakończenia już istniejącego, jest podstawową funkcją systemu. Usługa sieciowa nasłuchuje na adresie `/nfcTimeControl/service/reg` zgłoszeń przychodzących od aplikacji mobilnych.

Każde zgłoszenie przychodzące od telefonu komórkowego, zawiera dane pozwalające na uwierzytelnienie pracownika oraz dodatkowe informacje, takie jak identyfikator znacznika oraz data utworzenia zgłoszenia, na temat rozpoczętego czy też zakończonego zdarzenia. Dzięki nim system jest w stanie wprowadzić do bazy danych nowe zdarzenie, oznaczając je jednocześnie statusem „rozpoczęte”. W przypadku, gdy zgłoszenie ma na celu zakończenie wcześniej już zarejestrowanego zdarzenia przesyłany jest specjalnie tworzony token jednoznacznie identyfikujący zdarzenie. Na jego podstawie zdarzenie uzyskuje status „zakończone”. Token jest tworzony w oparciu o nazwę użytkownika zajmującego się zdarzeniem, datę utworzenia oraz identyfikator znacznika.

Uproszczony schemat działania rejestracji nowego zdarzenia został przedstawiony na rysunku 5.2.



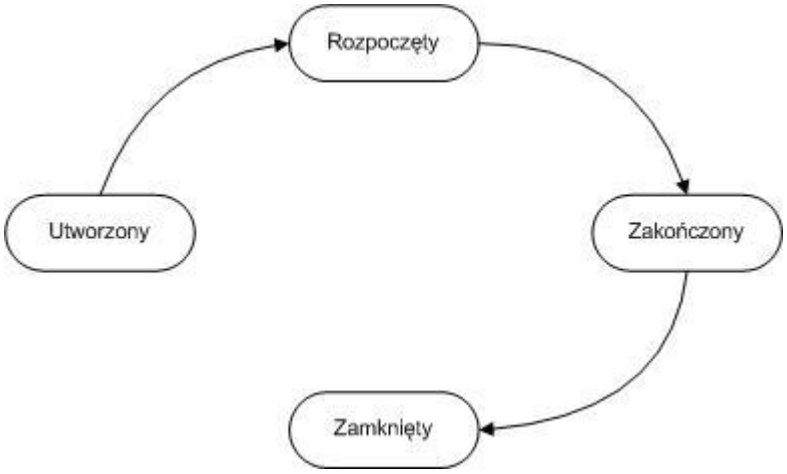
Rys. 5.2 Algorytm obsługujący zgłoszenie rejestracji zdarzenia

Na powyższym schemacie, widać krok po kroku jakie czynności wykonuje system przez odnotowaniem zdarzenia. Zaraz po uwierzytelnieniu użytkownika aplikacja sprawdza czy w przychodzącym zgłoszeniu przesłany jest token. Jeżeli jest, wiemy, że zdarzenie jest już w statusie „rozpoczęte” w bazie danych i należy je zakończyć poprzez ustawienie odpowiedniego statusu, a także daty zakończenia.

W przypadku braku tokenu, system przystępuje do kolejnego kroku jakim jest sprawdzenie czy dany identyfikator znacznika jest zarejestrowany w systemie. Jeżeli weryfikacja jest negatywna, aplikacja mobilna otrzymuje odpowiedni status błędu mówiący o braku znacznika w bazie. W przeciwnym razie aplikacja sprawdzi najpierw czy istnieją zdarzenia o statusie „utworzone” dla danego znacznika, lub też utworzy nowe zdarzenie.

Każdy krok kończy się przesłaniem odpowiednio sformułowanej odpowiedzi będącej klasą `CResponse`, która to zawiera pole informujące o statusie przetwarzanego zgłoszenia oraz dwa pola wypełniane w przypadku rejestracji zgłoszenia – komentarz do zgłoszenia utworzonego wcześniej poprzez panel administracyjny oraz token.

Na poniższym schemacie przedstawiono cykl życia nowo zarejestrowanego zdarzenia.



Rys. 5.3 Cykl życia zdarzenia

Wprowadzanie nowej lokalizacji i znacznika

Funkcja rejestracji nowej lokalizacji jest przeznaczona dla pracowników posiadających odpowiednio wysokie uprawnienia, takie jak `ROLE_SUPER_USER` czy też `ROLE_ADMIN`.

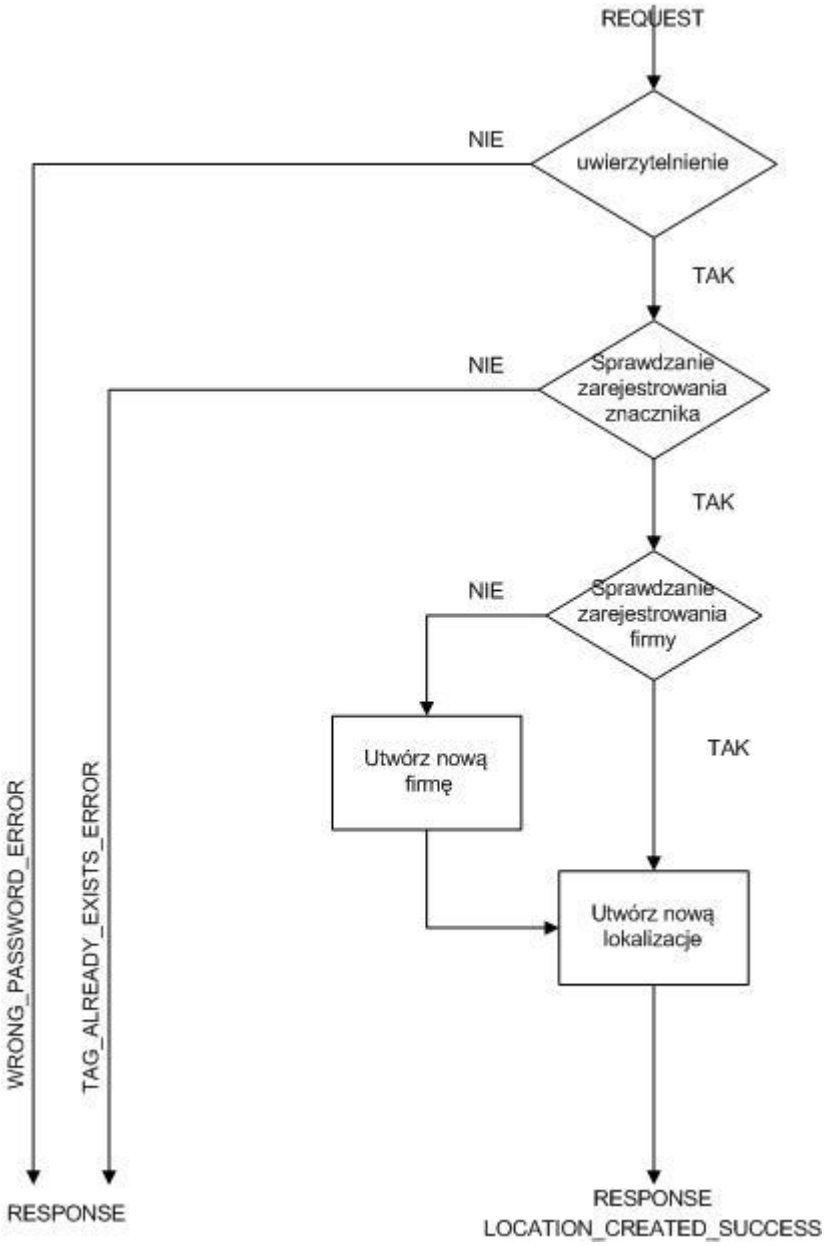
Mechanizm ten został stworzony z myślą o potrzebie łatwej rejestracji znaczników bezpośrednio na miejscu w zdalnej lokalizacji, na przykład podczas podpisywania nowej umowy z kontrahentem. Umożliwia ona osobie wprowadzającej nową lokalizację wprowadzenie podstawowych danych do zapisania jej w systemie.

Tak jak w poprzednich przypadkach, usługa sieciowa obsługuje zgłoszenia przychodzące na adres *`/nfcTimeContorl/service/admin`*. Każde zgłoszenie zawiera informacje na temat użytkownika służące do jego uwierzytelnienia, oraz identyfikator znacznika, datę rejestracji, nazwę lokalizacji, numer seryjny urządzenia z którym będzie powiązany znacznik, a także NIP i nazwę firmy do której należy dana lokalizacja. Ponieważ w sposób jednoznaczny identyfikuje on firmę, Numer Identyfikacji Podatkowej jest w systemie stosowany jako unikatowy identyfikator kontrahenta.

Na podstawie przesłanych danych, aplikacja decyduje o swoich następnych krokach. Jeżeli identyfikator znacznika:

- istnieje w bazie – usługa odmawia jego rejestracji
- nie istnieje, ale firma będąca jej właścicielem istnieje – rejestruje nową lokalizację
- nie istnieje, oraz firma, do której należeć będzie lokalizacje również nie istnieje – rejestruje najpierw nową firmę a następnie lokalizację.

Uproszczony mechanizm działania tej funkcjonalności przedstawia rysunek 5.4



Rys. 5.4 Algorytm rejestrowania nowej lokalizacji

Ze schematu można odczytać jak wygląda algorytm tworzenia nowej lokalizacji. Na początku zostaje potwierdzona tożsamość użytkownika. Następnie system sprawdzi czy dany znacznik istnieje w bazie danych. Jeżeli istnieje, użytkownikowi zostanie zwrócony odpowiedni status. W przeciwnym wypadku, baza danych zostanie odpytana czy dla przychodzącego w zgłoszeniu numeru NIP, istnieje firma zarejestrowana w systemie. Jeżeli tak, nowa lokalizacja zostanie stworzona i powiązana z istniejącą już firmą. W przeciwnym wypadku system zarejestruje nową firmę, lokalizację, a następnie zwróci użytkownikowi odpowiedni status opisujący pomyślne zakończenie procesu.

5.3. OPIS APLIKACJI KLIENCKIEJ NA TELEFON KOMÓRKOWY

5.3.1. Cel powstania

Aplikacja kliencka pełni funkcję intuicyjnego i prostego w obsłudze interfejsu do systemu monitorowania stanu realizacji zleceń.

5.3.2. Założenia

Główne założenia które stanowiły podstawę przy projektowaniu tej części systemu to możliwie jak największa szybkość wywołania i działania aplikacji, energooszczędność, łatwość w obsłudze i przejrzysta dla użytkownika prezentacja danych.

Uproszczony opis zadań powierzonych aplikacji na telefon zawiera w sobie:

1. automatyczny odczyt zbliżonego znacznika w sytuacji w której aplikacja jest włączona
2. zgłoszenie się do odczytu znacznika zgodnego z aplikacją z poziomu menu systemu operacyjnego
3. uwierzytelnienie użytkownika i automatyzacja logowania z zachowaniem oczekiwanej kontroli tożsamości
4. komunikację z użytkownikiem w oparciu o graficzny interfejs
5. dużą szybkość reakcji na działania użytkownika
6. komunikację z serwerem w celu obsługi odczytanego znacznika

W celu maksymalnego możliwego odciążenia aplikacji na telefon, zgodnie z ideologią pisania aplikacji na powyższy system, cała logika biznesowa zrealizowana jest w aplikacji serwerowej, telefon z kolei formułuje zapytania i przetwarza odpowiedzi. Komunikacja w architekturze REST (ang. *Representational State Transfer*) opiera się o zapytania w formie JSON (ang. *JavaScript Object Notation*). Bezpośrednio oznacza to, że aplikacja zgłasza do serwera zapytanie w momencie gotowości do przejścia w inny stan działania (np. przedstawienie obecnego stanu zgłoszenia). Forma tego zapytania jest ściśle znana i wysyłana w formacie JSON. Zaletą tego formatu jest ogólnodostępność i niezależność od języka programowania [15].

5.3.3. Komunikacja pomiędzy usługą internetową a użytkownikiem aplikacji

Na potrzeby komunikacji między aplikacją mobilną i usługą internetową stworzony został zestaw statusów przenoszonych w obiekcie `CResponse` odpowiedzi. Do zadań aplikacji należy możliwie jak najbardziej nieingerencyjne obsłużenie danego statusu, lub poinformowanie użytkownika o zaistniałej sytuacji.
Do sytuacji takich należą między innymi:

- komunikaty o braku dostępu:
  `ACCESS_DENIED`,
  `WRONG_PASSWORD_ERROR`,
  `USER_NOT_FOUND_ERROR`
- komunikat o nie odnalezieniu danego znacznika w bazie danych:
  `TAG_NOT_FOUND_ERROR`
- komunikat informujący o fakcie uprzedniego rozpoczęcia realizacji innego zgłoszenia (przy próbie rozpoczęcia innego):
  `EVENT_ALREADY_STARTED_ERROR`

W każdej z tych sytuacji aplikacja informuje użytkownika i dokonuje przekierowania interfejsu w celu ułatwienia rozwiązania sytuacji (przejście do ekranu logowania, wyświetlenie stanu obecnie wykonywanego zlecenia itp.)

5.3.4. Tryby uruchomienia

Z przyczyn praktycznych aplikacja może być wywołana dwojako: przez użytkownika z poziomu menu systemu operacyjnego Android, lub automatycznie w wyniku zbliżenia telefonu do znacznika. Metody te różnią się cyklem wywołań intencji i działaniami wykonywanymi w tle.

5.3.4.1. Uruchomienie z poziomu menu

W pierwszym wypadku do obsługi znacznika stosowany jest tzw. *foreground dispatching*, opisany dokładniej w podrozdziale 5.4.6. Niestety metoda ta nie działa zawsze idealnie i czasem system operacyjny pomimo tego faktu pyta się użytkownika o to której aplikacji użyć do obsłużenia danego znacznika, jednak w obecnej wersji aplikacji są to przypadki sporadyczne i nie wpływają na działanie aplikacji.

Dodatkowo aplikacja w pierwszej fazie uruchomienia sprawdza czy posiada zapisane w banku *SharedPreferences* dane logowania i jeśli nie może ich znaleźć, oferuje użytkownikowi możliwość wprowadzenia ich bądź wyjścia z aplikacji [18]. Ponieważ przy każdym odczytaniu znacznika dane zdarzenia transmitowane do serwera opatrzone są

kompletem danych identyfikacyjnych (z racji przyjętej architektury aplikacji typu RESTful system nie utrzymuje sesji użytkowników tylko wymienia zapytania i odpowiedzi), w sytuacji zgłoszenia jednego ze statusów braku dostępu, zachodzi przekierowanie do aktywności wprowadzania danych logowania.

Zmiana danych logowania jest dostępna z poziomu menu aplikacji w każdej chwili. Hasło do logowania przechowywane jest w aplikacji tylko w formie skrótu *SHA-1* w celu uniemożliwienia przechwycenia jego jawnej postaci.

W tym wariancie uruchomienia aplikacja dokonuje również synchronizacji z usługą internetową w celu pozyskania stanu obecnie obsługiwanego przez danego użytkownika zdarzenia, jeśli takie istnieje. Dzięki temu nawet w sytuacji odinstalowania i ponownego zainstalowania, zmiany danych logowania lub usunięcia danych użytkownika jest on zawsze poinformowany o swoim stanie zarejestrowanym w systemie.

5.3.4.2. Uruchomienie poprzez zbliżenie telefonu do znacznika

Sytuacją odmienną jest zbliżenie telefonu do znacznika w stanie spoczynku (przy wyłączonej aplikacji).

W sytuacji takiej aplikacja uruchamia się tylko w celu odnotowania zdarzenia i po zakończeniu tego procesu i poinformowaniu użytkownika o efekcie automatycznie się zamyka (nie wymaga to interakcji ze strony użytkownika i nie zajmuje pamięci systemu). Oczywiście w sytuacji podania niepoprawnych danych logowania, interfejs wyświetla stosowny komunikat i prowadzi do aktywności wprowadzenia danych użytkownika.

5.3.5. Odczyt znacznika

Standard znaczników typu NFC Forum Type 2 posiada jedynie 384 bitów pamięci dostępnych do przechowywania danych użytkownika [20]. Po przeprowadzeniu szeregu testów ustalono, że zapis szczegółowych danych na znaczniku jest niepożądany z punktu widzenia założeń systemu. Chcąc zapewnić pełną integralność danych komplet informacji przechowywany jest w bazie danych. Tym samym unikając możliwości ich udostępnienia osobom trzecim.

System operacyjny w momencie wykrycia znacznika w obrębie pola elektromagnetycznego wytwarzanego przez telefon pobiera jego unikatowy identyfikator oraz wiadomości NDEF i umieszcza w intencji, która w dalszej kolejności przekazywana jest do odpowiedniej aplikacji w celu ich przetworzenia [18]. Spośród tych danych aplikacja wykorzystuje jedynie unikatowy identyfikator znacznika, który przesyłany jest do bazy danych.

5.3.6. Dobór aplikacji do obsługi znacznika

W wersji systemu operacyjnego Android Gingerbread 2.3.4 komunikacja NFC doczekała się serii modyfikacji i rozszerzeń które pozwoliły między innymi w bardziej precyzyjny sposób delegować odpowiednią aplikację do obsługi danych zawartych na znaczniku [9]. W tym celu stworzono dwa systemy wywoływania aplikacji.

5.3.6.1. Intent dispatch

Pierwszym z nich jest Intent dispatch składający się z trzech następujących po sobie filtrów *akcji*, które mają na celu możliwie jak najbardziej zawężać pulę zainstalowanych aplikacji tak, aby użytkownik telefonu nie musiał podejmować żadnych decyzji tylko był obsługiwany automatycznie. Jeśli dane kryterium spełnia więcej niż jedna aplikacja, wyświetlany jest monit z prośbą o wybór odpowiedniej [11]. Poniżej zostaną przedstawione wspomniane typy akcji:

`android.nfc.action.NDEF_DISCOVERED`

Powyższa akcja opiera się na analizie typu wiadomości zamieszczonych na znaczniku. Jeśli dany typ wiadomości widnieje w pliku `AndroidManifest.xml` dowolnej aplikacji, na tym kroku przerywana jest analiza i podejmowany wybór

`android.nfc.action.TECH_DISCOVERED`

W sytuacji gdy poprzedni filtr zawiedzie, sprawdzany jest technologia w jakiej wykonany został znacznik.

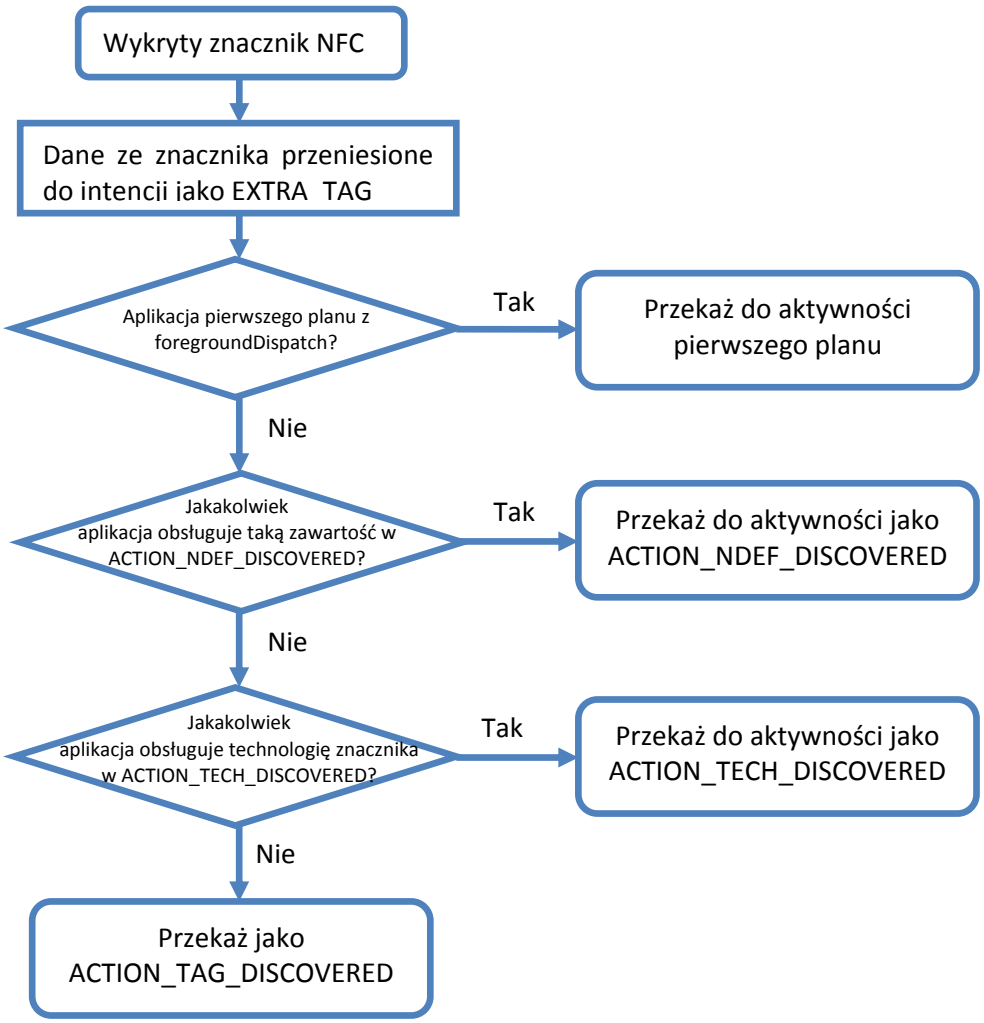
`android.nfc.action.TAG_DISCOVERED`

Akcja ta wywoływana jest w sytuacji gdy powyższe filtry zawiodły, lub znacznik jest nieznanego typu.

5.3.6.2. Foreground dispatch

Druga z metod obsługi odczytanego znacznika, tzw. Foreground dispatch, wymaga od programisty samodzielnego wywołania i kontroli w cyklu życia aplikacji. Opiera się ona na narzuceniu najwyższego priorytetu obsługi znacznika aplikacji uruchomionej na pierwszym planie [18]. Oczywiście znacznik spełniać musi wymogi określone w filtrach typu danych i technologii jego wykonania.

Jak było wspomniane wcześniej, aplikacja kliencka wykorzystuje obie te metody. Diagram poniżej przedstawia uproszczony schemat działania połączonych obu wspomnianych systemów.



Rys. 5.5 Proces podejmowania decyzji o metodzie obsługi znacznika

5.3.7. Wprowadzenie danych i komunikacja z usługą internetową

Weryfikacja autentyczności użytkownika nie może przebiegać z użyciem samej nazwy użytkownika i hasła. Do grupy danych które celem uwierzytelnienia musi przesłać aplikacja kliencka dodane zostały numer IMEI (International Mobile Equipment Identity) oraz IMSI (International Mobile Subscriber Identity). Dzięki temu aplikacja serwerowa weryfikuje, czy użytkownik o danej nazwie użytkownika zgłasza swoją aktywność przy pomocy swojego telefonu (numer IMEI) oraz swojej karty SIM (numer IMSI). To założenie eliminuje możliwości oszukania systemu płynące z wymiany telefonami i danymi autoryzacyjnymi pomiędzy jego użytkownikami.

Parametr	Opis
Identyfikator użytkownika	Przydzielony przez administratora systemu, unikatowy w skali systemu
Hasło	Spełniające wymogi bezpieczeństwa[4]
Numer IMEI	Unikatowy, piętnasto lub siedemnastocyfrowy identyfikator urządzenia GSM lub UMTS, nie jest związany z konkretnym użytkownikiem.
Numer IMSI	Unikatowy numer identyfikujący kartę SIM. Stworzony na potrzeby identyfikacji abonenta na drodze radiowej w systemach GSM i UMTS.

Tabela 5.4 Dane brane pod uwagę w procesie uwierzytelnienia

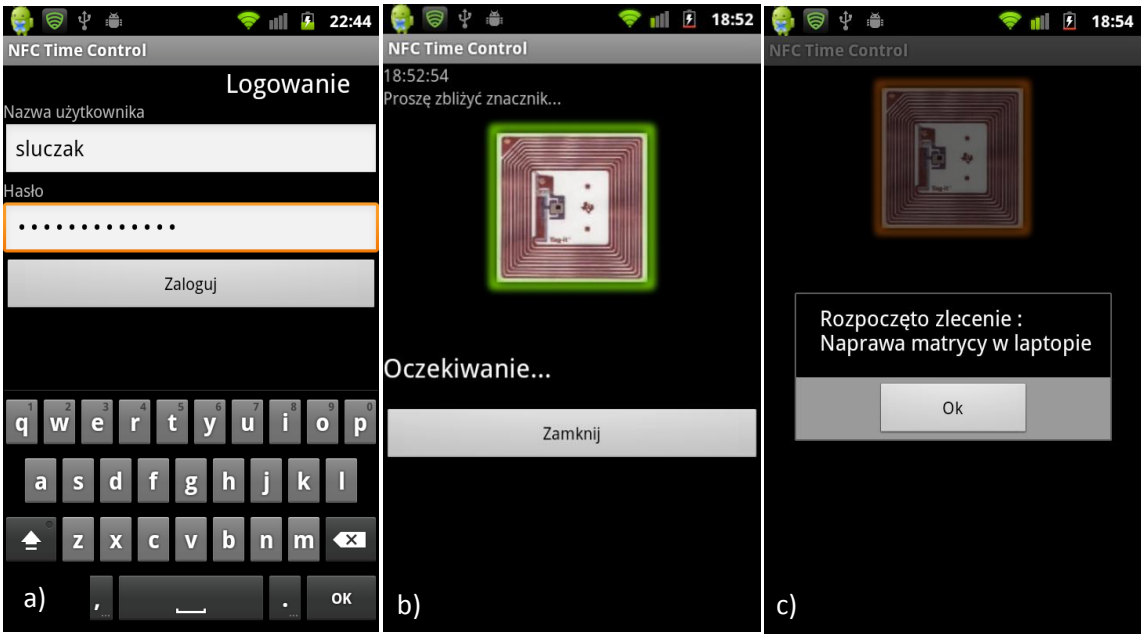
Powyższy komplet danych identyfikuje użytkownika i telefon w sposób jednoznaczny. Numery IMSI i IMEI nie są zapisywane nigdzie w pamięci ani ustawieniach aplikacji, tylko pobierane poprzez API systemu operacyjnego każdorazowo przy uruchomieniu aplikacji.

5.3.8. Działanie aplikacji

Po pierwszym uruchomieniu aplikacji użytkownik musi wprowadzić nazwę użytkownika i hasło w celu zalogowania w systemie. Jeśli proces uwierzytelnienia przebiegnie pomyślnie, dane zostaną zapisane w banku danych aplikacji systemu Android i będą od teraz używane do rejestracji zdarzeń w systemie aż do ich zmiany lub usunięcia. Wspomniane opcje dostępne są zawsze z poziomu menu podręcznego.

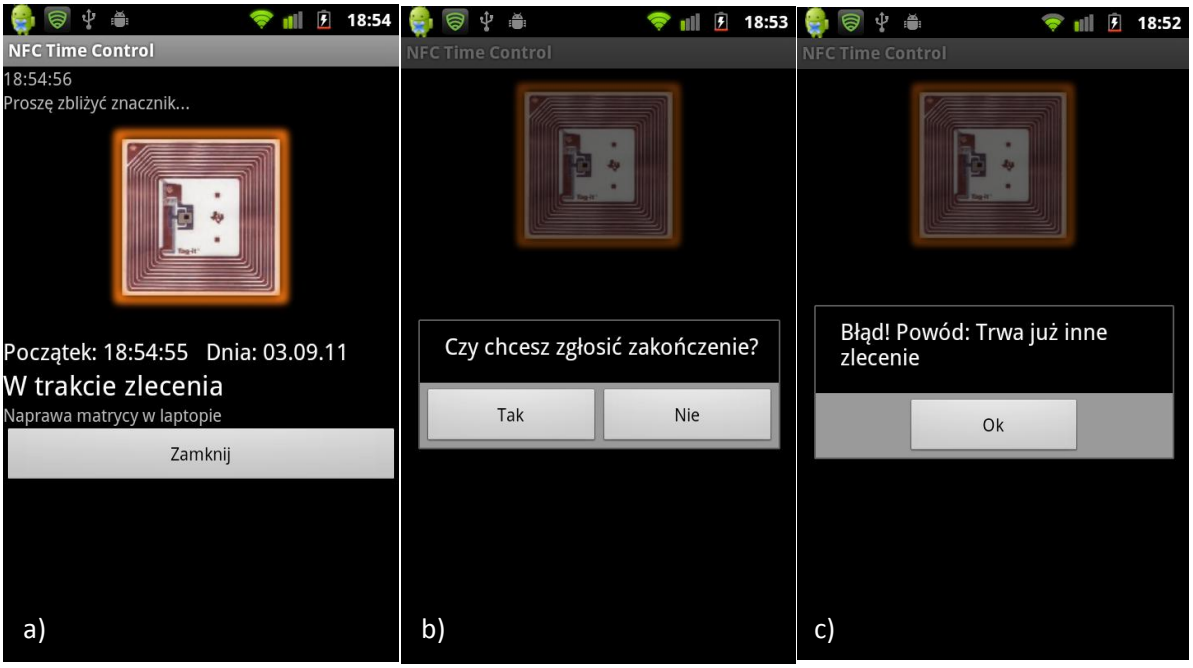
Główny ekran aplikacji wyświetla tekstowo i graficznie obecny stan aktywności danego użytkownika. Jest to „Odświeżanie…” jeśli informacje z serwera jeszcze nie dotarły, „W trakcie zlecenia…” lub „Oczekiwanie…” w sytuacji w której żadne zlecenie nie jest obecnie obsługiwane. Jeśli zlecenie zostało rozpoczęte, dodatkowo wyświetlany jest krótki komentarz wprowadzony przez operatora który wprowadził dane zlecenie do systemu oraz data i godzina jego rozpoczęcia. Informację o stanie użytkownika aplikacja pozyskuje w procesie synchronizacji z usługą internetową tuż po uruchomieniu aplikacji.

[4] 8 znaków, duże i małe litery



Rys. 5.6 a) Ekran logowania aplikacji b) Ekran oczekiwania c) Rozpoczęcie zlecenia

W sytuacji chęci rozpoczęcia zlecenia, użytkownik musi jedynie zbliżyć telefon do znacznika. Na podstawie kompletu danych użytkownika, identyfikatora i daty systemowej formułowane jest zapytanie wysyłane do usługi internetowej. Data systemowa wysyłana jest w celu porównania z datą aplikacji serwerowej w sytuacji w której pracownik mógłby mieć zastrzeżenia co do stanu rozliczenia z pracodawcą. W wiadomości zwrotnej przekazywany jest unikatowy token danego zdarzenia, niezbędny przy zgłaszaniu jego zakończenia. Rozpoczęcie zdarzenia sygnalizowane jest wibracją, dźwiękiem i odpowiednim oknem dialogowym, zawierającym komentarz wprowadzony wcześniej przez operatora.



Rys. 5.7 a) Ekran statusu b) Kończenie zlecenia c) Błąd związany z próbą rozpoczęcia drugiego zlecenia

Z poziomu ekranu opcji istnieje możliwość wyłączenia wszystkich metod sygnalizacji prócz okna dialogowego.

Zakończenie realizacji zlecenia wykonuje się analogicznie, z tym że wymaga to również potwierdzenia poprzez stosowne okno dialogowe.

Usiłowanie rozpoczęcia realizacji drugiego zlecenia przed zamknięciem poprzedniego zostanie zauważone przez aplikację serwerową i odrzucone z odpowiednim statusem. Użytkownik zostanie poinformowany o tym fakcie poprzez stosowne okno dialogowe. Również w sytuacji wystąpienia błędu komunikacji lub wyjątku, użytkownik zostanie o tym poinformowany, a aplikacja powróci do stanu oczekiwania.

Wszystkie procesy komunikacji z usługą internetową przeniesione są do wątku roboczego, toteż problemy z połączeniem lub wydłużony czas realizacji zapytania nie wpływa na interfejs graficzny, dzięki czemu nawet przy wystąpieniu problemów z zasięgiem nie dochodzi do zjawiska tzw. braku odpowiedzi ze strony aplikacji ANR (ang. *Application Not Responding*).

5.4. OPIS APLIKACJI ADMINISTRACYJNEJ NA TELEFON KOMÓRKOWY

5.4.1. Cel powstania

Aplikacja administracyjna używana jest do rejestracji lokalizacji usługowych przez operatora lub administratora systemu. Za jej pomocą wiąże on dany znacznik z klientem i konkretnym urządzeniem. Wywoływana jest przez użytkownika z poziomu systemu operacyjnego.

5.4.2. Główne założenia

Głównym założeniem przyświecającym jej powstaniu było możliwie jak największe przyśpieszenie procesu rejestracji znaczników bezpośrednio u kontrahenta.

5.4.3. Działanie aplikacji

Po uruchomieniu aplikacji użytkownik proszony jest o wprowadzenie nazwy użytkownika i hasła. Przy procesie autoryzacji, prócz zestawu danych wspomnianych w rozdziale opisującym aplikację kliencką, weryfikowana jest również rola przydzielona użytkownikowi w systemie. Użytkownicy typu `ROLE_USER` nie mogą używać tej aplikacji i ich próby są odrzucane przez system. Po poprawnym uwierzytelnieniu, dane użytkownika zostają zapisane w systemie, można je jednak zmienić lub wyczyścić korzystając z menu podręcznego.

Poprawna autoryzacja przenosi użytkownika do ekranu wprowadzania danych rejestracji znacznika. W formularzu rejestracyjnym udostępnione są pola za pośrednictwem których użytkownik musi wprowadzić nazwę urządzenia, jego numer seryjny, a także skrótową nazwę kontrahenta i jego numer NIP. Pola formularza podlegają walidacji, a skrótowa nazwa kontrahenta brana jest pod uwagę jedynie w sytuacji w której ma miejsce rejestracja nowego klienta.

Jak widać, od użytkownika oczekiwana jest wprowadzenie minimalnej liczby danych. Jest tak, ponieważ wprowadzanie tekstu na telefonie komórkowym może być uciążliwe, co naraża dane na występowanie dużej liczby błędów. W związku z powyższym, interfejs użytkownika używany jest jedynie do przekazania danych niezbędnych do jednoznacznego skojarzenia znacznika, urządzenia i kontrahenta, a reszta pól uzupełniana jest przez operatora za pośrednictwem portalu administracyjnego z poziomu komputera.

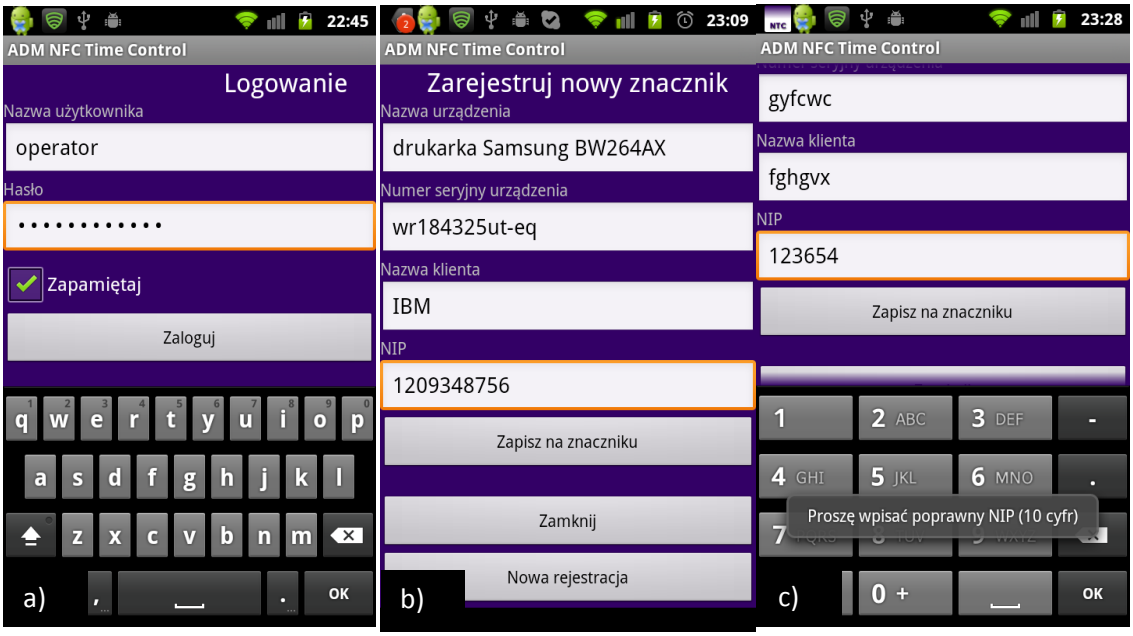
Ponieważ często może mieć miejsce rejestracja wielu urządzeń u jednego kontrahenta, pole zawierające NIP nie jest czyszczone w jednej sesji działania aplikacji, tylko po jej wyłączeniu i ponownym włączeniu. Dzięki temu proces ten podlega kolejnemu uproszczeniu. Wybranie przycisku „Zapisz na znacznik” rozpoczyna walidację formularza.

Po pomyślnym przejściu walidacji pól, aplikacja zwraca się do użytkownika z prośbą o zbliżenie znacznika i korzystając z systemu Foreground dispatch dostrojonego tak, aby reagował tylko na znaczniki typu Mifare Ultralight, uzyskuje najwyższy priorytet przy jego obsłudze [11].

W następnym kroku, jeśli użytkownik zbliży znacznik, aplikacja odczytuje jego unikatowy identyfikator, wysyła zapytanie do usługi internetowej i jeśli operacja przebiegnie pomyślnie, blokuje ponowną możliwość zapisu. Początkowo aplikacja ta prócz wiązania urządzenia i kontrahenta z identyfikatorem znacznika, zapisywała również część tych danych na znaczniku. W trakcie testów okazało się jednak, że jest to funkcja nadmiarowa i nie poprawia działania systemu tylko go komplikuje. Wniosek taki wynika z faktu, iż ograniczony rozmiar pamięci znacznika nie pozwalał zapisać kompletu danych, a często nawet jednego długiego pola. Opieranie zatem weryfikacji danych na zawartości znacznika było zawodne Z tego powodu rejestracja kompletu danych odbywa się po stronie usługi internetowej, a na znaczniku zapisywana jest tylko nazwa aplikacji. Zapis ten, choć również wydaje się zbędnym, powodowany jest przyczynami technicznymi. Jak zostało wspomniane w rozdziale opisującym aplikację kliencką, system *Foreground dispatch* pierwszą filtrację aplikacji do obsłużenia znacznika wykonuje na podstawie typu danych przechowywanych przez znacznik. Jedną z możliwości jest określenie typu danych MIME na jaki ma reagować

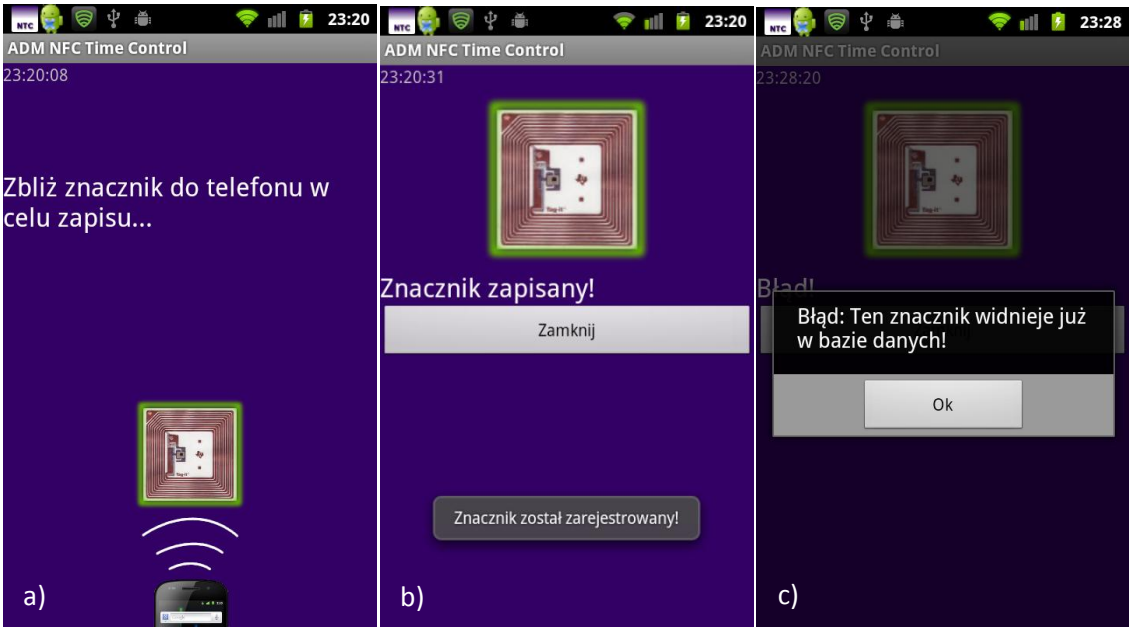
dana aplikacja. Ponieważ typy te można niezwykle łatwo deklarować, na znaczniku zapisywana jest nazwa aplikacji przy użyciu własnego typu `text/vnd.timecontrol`.

W efekcie jedyną aplikacją która zgłasza się do obsłużenia tego znacznika jest aplikacja kliencka prezentowanego systemu. Skraca to czas wywołania i powoduje, że użytkownik nie musi ręcznie wybierać aplikacji z listy.



5.8 a) Logowanie b) Rejestracja znacznika c) Wprowadzenie błędnych danych

Po pomyślnej rejestracji znacznika, za pośrednictwem interfejsu użytkownik otrzymuje stosowny komunikat i może kontynuować rejestrację pozostałych urządzeń. W przeciwnym wypadku system powiadamia o niepowodzeniu i jego przyczynie oraz odsyła do poprzedniego ekranu.



5.9 a) Foreground dispatch – nasłuchiwanie na znacznik b) Pomyślny zapis znacznika c) Wystąpienie błędu

Ponieważ duże powierzchnie metalowe zakłócają komunikację NFC, rejestrację znacznika należy wykonywać w bezpośredniej okolicy oklejanego urządzenia. Najlepiej wykonywać to przykładając znacznik do obudowy tak, aby ostatecznie został przyklejony w miejscu w którym komunikacja telefonu ze znacznikiem przebiega w sposób bezproblemowy.

5.5. OPIS PANELU ADMINISTRACYJNEGO

5.5.1. Cel powstania

Panel administracyjny jest ostatnim a zarazem najbardziej rozbudowanym elementem systemu. Pełni kluczową rolę w zarządzaniu rozwiązaniem.

Użytkownikami panelu administracyjnego są osoby zarządzające czasem pracy pracowników mobilnych. Dzięki dostarczonym funkcjom, mogą one bez trudu kontrolować czas oraz stopień zaawansowania pracy. Aplikacja udostępnia również wgląd do informacji zgromadzonych w bazie danych. Dzięki niej możliwe jest edytowanie wpisów oraz dodawanie nowych do już istniejących.

5.5.2. Szczegóły implementacyjne

Aplikacja webowa do poprawnego działania wymaga obecności dowolnego serwera aplikacyjnego zgodnego ze standardem Java Enterprise Edition. Musi być on kompatybilny z technologią Servletów oraz Java Server Pages. W przypadku opisywanego systemu, użyty został serwer aplikacyjny Apache Tomcat w wersji 6.0.32. Dzięki niemu możliwe było stworzenie warstwy prezentacji opartej na stronach JSP współpracujących z bibliotekami JSTL (ang. *Java Standard Tag Library*). Wykorzystując to rozwiązanie, możliwe jest oddzielenie kodu źródłowego od warstwy prezentowanych danych.

Środowiskiem programistycznym w którym został wykonany panel jest :

- Java SDK 1.6.0_26
- Tomcat 6.0.32
- Oracle 10g XE wraz z sterownikiem jdbc w wersji 6
- Eclipse Helios wraz z zainstalowanym pakietem Web Tools Platform

Dodatkowo panel administracyjny został wykonany w oparciu o szkielet aplikacji Spring w wersji 3.0.1, pakiet Spring Security w wersji 3.0.5 oraz technologię ORM (ang. *Object Relational Mapping*), którą dostarcza Hibernate 3. W celu zapewnienia poprawności

logowania zdarzeń aplikacji oraz ewentualnych wyjątków zawarty został również framework Log4J.

Aplikacja webowa implementuje wzorzec MVC – Model Widok Kontroler opisany we wcześniejszej części pracy. Poniżej znajduje się opis struktury stworzonego panelu.

5.5.3. Warstwa biznesowa

Warstwę biznesową tworzą moduły zawierające klasy biorące udział w tworzeniu logiki aplikacji. Są one odpowiedzialne za pośredniczenie pomiędzy warstwą prezentacji oraz warstwą modelu biznesowego. Dokonują obróbki i walidacji informacji wpisywanych przez użytkownika końcowego przed ich utrwaleniem w bazie danych. Warstwa ta posiada także mechanizmy które umożliwiają systemowi wysyłanie wiadomości elektronicznych do użytkowników portalu zawierających listę utworzonych zdarzeń wymagających dalszej reakcji.

Lista modułów została przedstawiona poniżej :

- *net.nfc.web.controller*
- *net.nfc.web.forms*
- *net.nfc.web.mailservice*
- *net.nfc.web.scheduler*
- *net.nfc.web.service*
- *net.nfc.web.validator*

net.nfc.web.controller zawiera klasy odpowiadające za odpowiednie mapowanie zapytań, przychodzących z warstwy prezentacji na odpowiednie zasoby. Zawarta jest w nich logika pozwalająca na przetwarzanie danych z formularzy oraz przesyłanie ich dalej do odpowiednich obiektów zajmujących się ich walidacją lub zapisem do bazy danych.

net.nfc.web.form grupuje klasy potrzebne do obsługi formularzy wyświetlanych na stronach JSP.

net.nfc.web.mailservice odpowiada za klasy potrzebne do obsługi wysyłki wiadomości elektronicznych, mających na celu informowanie pracowników o nowo utworzonych zdarzeniach.

net.nfc.web.scheduler jest odpowiedzialny za zarządzanie procesem wysyłki maili powiadamiających pracowników o nowo wprowadzonych do systemu zdarzeniach wymagających ich reakcji.

net.nfc.web.service jest modułem, który znajduję się na granicy pomiędzy warstwą modelu – czyli utrwalania danych, a warstwą biznesową. Jego zadaniem jest przesyłanie

obiektów do odpowiednich klas DAO (ang. *Data Access Object*) tak, aby te zapisały informacje w bazie danych.

net.nfc.web.validator zawiera klasy odpowiadające za poprawność danych przesyłanych w formularzach przez użytkownika końcowego. To jego zadaniem jest powstrzymanie przed wpisaniem do bazy niepoprawnych wartości czy też zadbanie o wypełnienie wszystkich obowiązkowych pól.

5.5.4. Warstwa modelu danych

Warstwa ta odpowiada za poprawne pobranie, uaktualnienie lub zapis obiektów do encji bazy danych. Za połączenie z bazą danych oraz zarządzenie transakcjami i sesjami odpowiada pakiet Hibernate. W skład tej klasy wchodzą dwa moduły :

- *net.nfc.db.dao*
- *net.nfc.db.entity*

Paczka *net.nfc.db.dao* zawiera klasy bezpośrednio kontaktujące się z bazą danych. Są w nich zawarte odpowiednie metody odpowiedzialne za pobieranie rekordów, listowanie danych, uaktualnianie bądź też wprowadzanie nowych informacji. Obiekty DAO również odpowiadają za filtrowanie wyników według odpowiednich kryteriów.

Moduł *net.nfc.db.entity* zawiera klasy mapowane na encje bazy danych. Są to podstawowe obiekty biznesowe reprezentujące np. pracownika, zdarzenie, lokalizację czy też kontrahenta. W paczce tej zawarte są dodatkowo pliki konfiguracyjne Hibernate, dzięki którym możliwa jest odpowiednia współpraca aplikacji z bazą danych.

5.5.5. Warstwa prezentacji

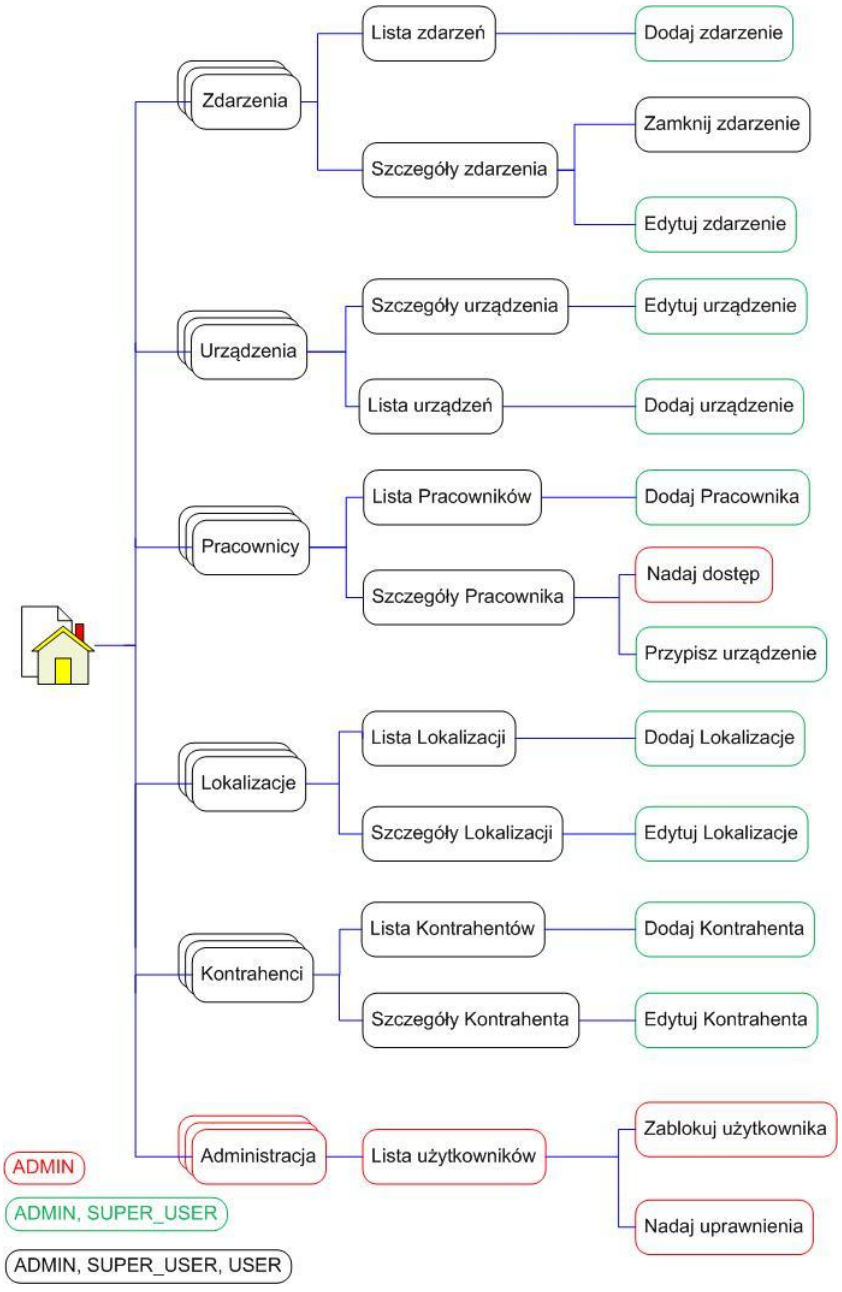
Warstwa prezentacji jest odpowiedzialna za poprawne wyświetlanie danych w ramach graficznego interfejsu użytkownika. Została ona wykonana za pomocą stron JSP wraz z użyciem biblioteki JSTL, języka HTML oraz kaskadowych arkuszy stylów CSS.

Warstwa prezentacji odgrywa istotną rolę w kontrolowaniu dostępu pracowników do funkcjonalności systemu. Są w niej zawarte mechanizmy niepozwalające na wyświetlenie bądź też edycję danych bez odpowiednich uprawnień.

Podczas jej tworzenia został położony nacisk na wyraźnie zarysowanie linii podziału interfejsu użytkownika od rdzenia funkcjonalnego systemu. Dzięki takiemu podejściu interfejs użytkownika może się łatwo zmieniać wraz z polepszaniem używalności systemu oraz wcielaniem nowych technologii do prezentacji danych.

5.5.6. Budowa panelu administracyjnego

Panel administracyjny został tak stworzony, aby jego obsługa była jak najbardziej intuicyjna dla użytkownika końcowego. Poniżej został zamieszczony schemat obrazujący budowę i funkcje panelu administracyjnego. Na rysunku przedstawiono mapę serwisu wraz z zaznaczonymi uprawnieniami niezbędnymi do uzyskania dostępu oraz wykonania danej akcji.



Rys. 5.10 Schemat budowy panelu administracyjnego

5.5.7. Role użytkowników oraz zabezpieczenie portalu

Za bezpieczeństwo portalu administracyjnego odpowiada pakiet Spring Security. Dba on o poprawne uwierzytelnienie użytkowników logujących się do panelu oraz za kontrolę dostępu do określonych zawartości panelu. Dzięki swoim zaawansowanym funkcjom dba o to aby jeden użytkownik mógł być zalogowany tylko raz do systemu. Dodatkowo zaimplementowana jest funkcja, która po dłuższym okresie bezczynności automatycznie wyloguje użytkownika.

Jak już wcześniej zostało opisane, każdy użytkownik logujący się do portalu posiada z góry przydzieloną rolę. Dzięki takiemu rozwiązaniu można określić poziom dostępu pracownika do danych wg poniższej schematu:

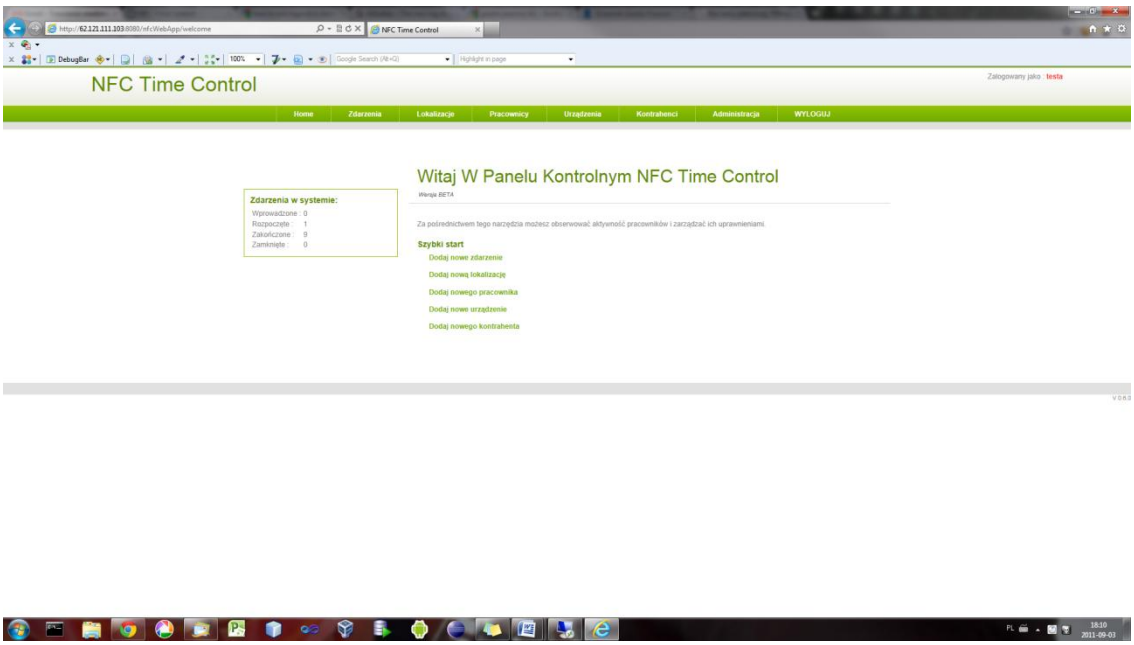
`ROLE_USER` – posiada wgląd do danych zgromadzonych w bazie na zasadzie read-only. Użytkownik z tym uprawnieniem może również edytować zdarzenie, którym się zajmował. Modyfikacja taka polega na ustawieniu go w statusie „zamknięte” i dodaniu opcjonalnego, dodatkowego komentarza.

`ROLE_SUPER_USER` – jest to rola przeznaczona dla operatora panelu. Operator ma możliwość edycji prezentowanych danych, tworzenia nowych zleceń, dodawania nowych kontrahentów, lokalizacji, pracowników oraz urządzeń.

`ROLE_ADMIN` – jest to uprawnienie przypisane administratorowi systemu. Posiada on te same uprawnienia co użytkownik z rolą `ROLE_SUPER_USER` i dodatkowo ma udostępnioną zakładkę „Administracja”, w której wyświetlani są wszyscy zarejestrowaniu użytkownicy do systemu. Istnieje możliwość odblokowania/zablokowanie użytkownika, oraz nadania mu odpowiednich uprawnień. Jego przywilejem jest także nadawanie dostępu do panelu i przydzielanie urządzeń pracownikom.

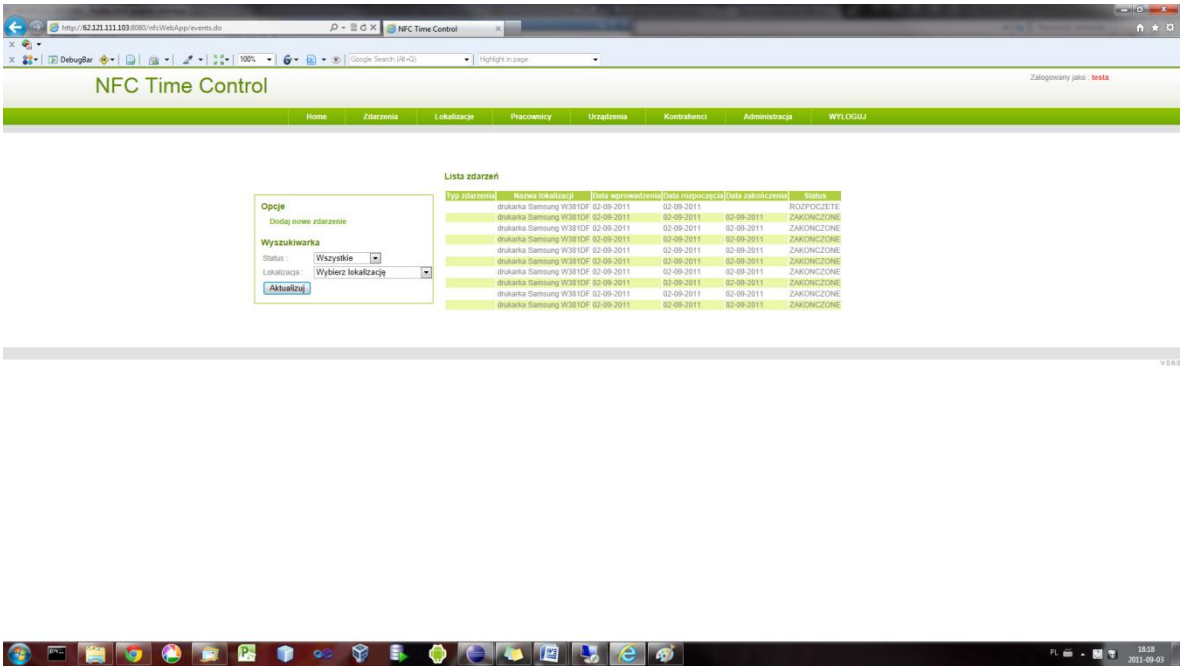
5.5.8. Opis zakładek

Użytkownik po zalogowaniu się do systemu zostaje przeniesiony do strony startowej, z której ma możliwość przejścia do dalszych części panelu.



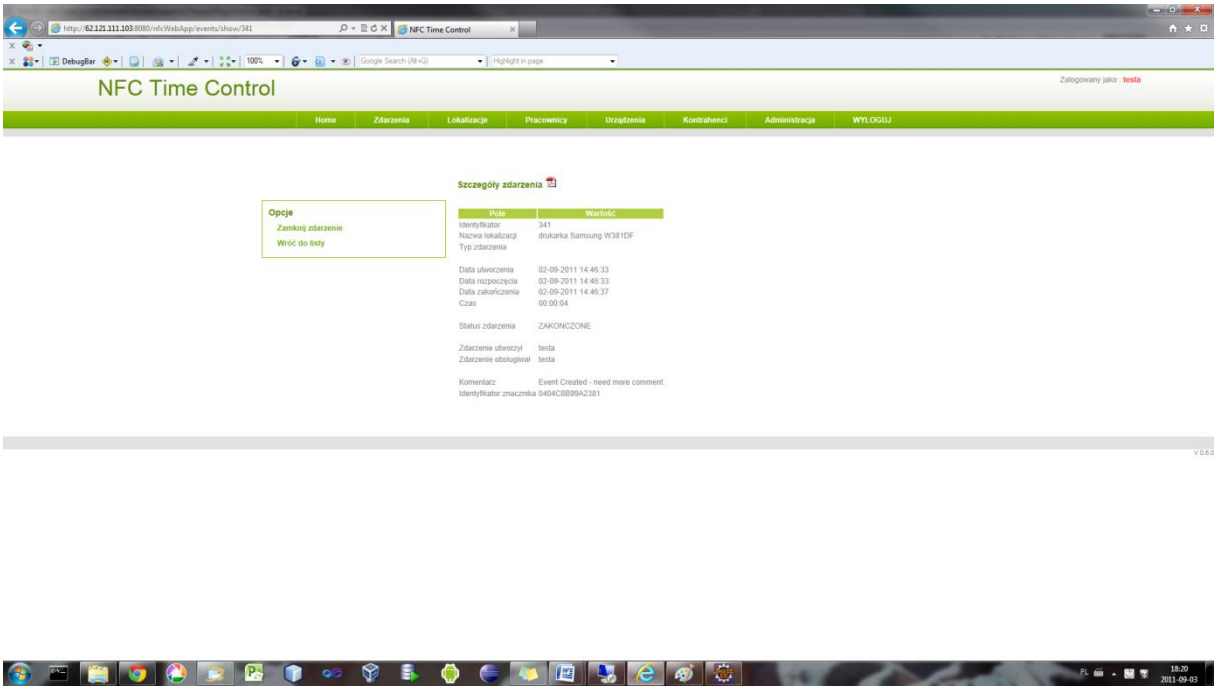
Rys. 5.11 Panel, strona startowa

Po wybraniu zakładki „Zdarzenia” operator zostaje przeniesiony do strony wyświetlającej wszystkie zdarzenia zarejestrowane w systemie. Możliwe jest wybranie kryteriów filtru w celu zawężenia wyświetlanych wyników oraz dodanie nowego zdarzenia do systemu. W takim przypadku, pracownik zostaje przeniesiony do formularza, przez który wprowadzi typ zdarzenia, komentarz oraz wybierze lokalizację usługi.



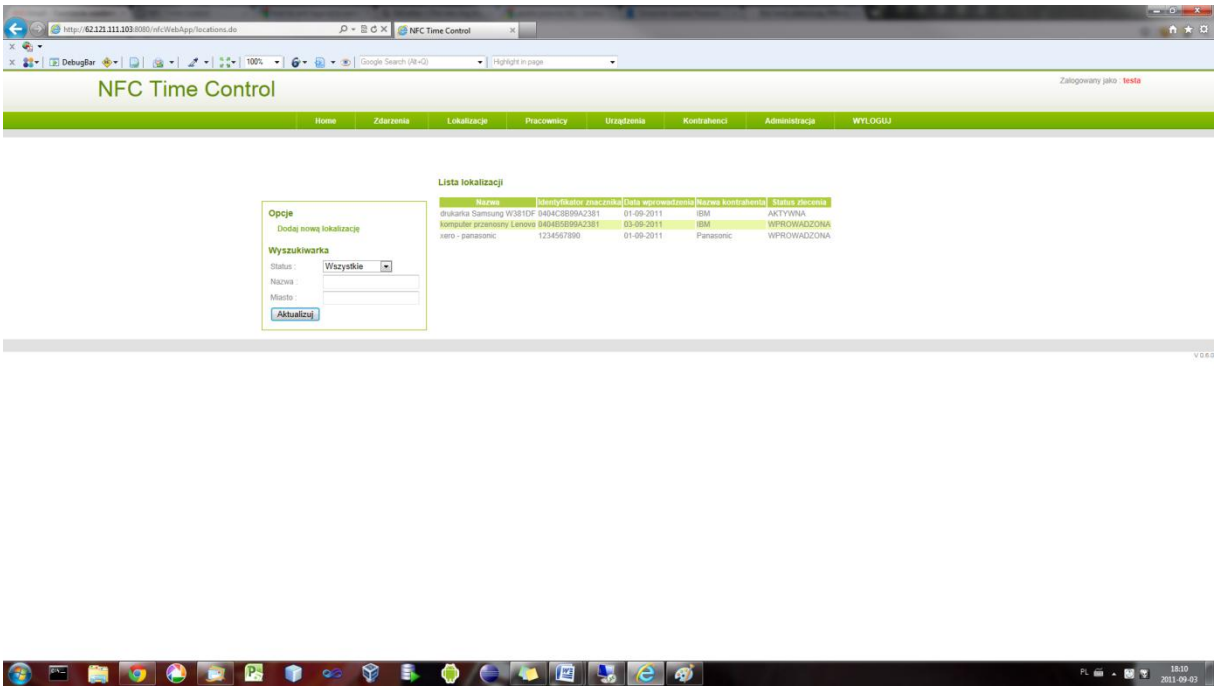
Rys. 5.12 Widok zakładki "Zdarzenia"

Każdy wyświetlany na liście wynik jest hiperłączem do zakładki wyświetlającej szczegółowe dane o danej pozycji z możliwością ich edycji. Operator ma możliwość wyeksportowania szczegółowych danych zdarzenia do pliku PDF.



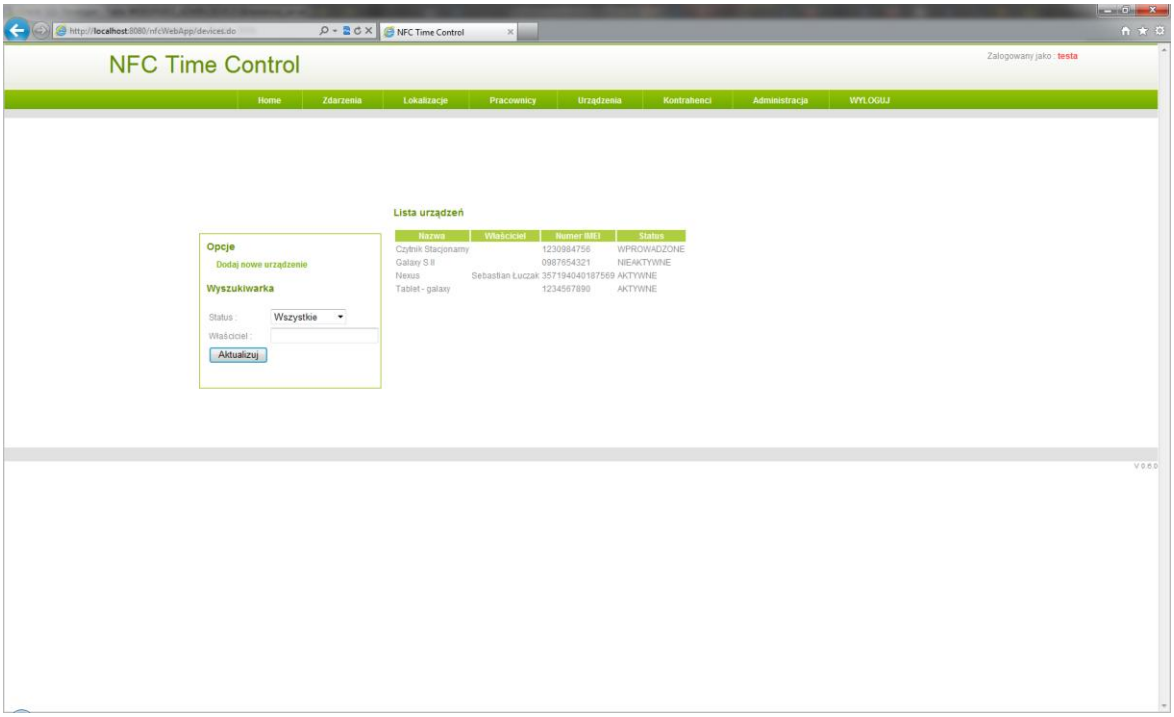
Rys. 5.13 Szczegółowy widok z danymi zdarzenia

Zakładka „Lokalizacje” przedstawia użytkownikowi listę wszystkich lokalizacji wprowadzonych do systemu. Pozwala na wgląd do szczegółowych danych o wybranej lokalizacji. Operator ma możliwość ich edycji jak i dodawania nowej pozycji do systemu.

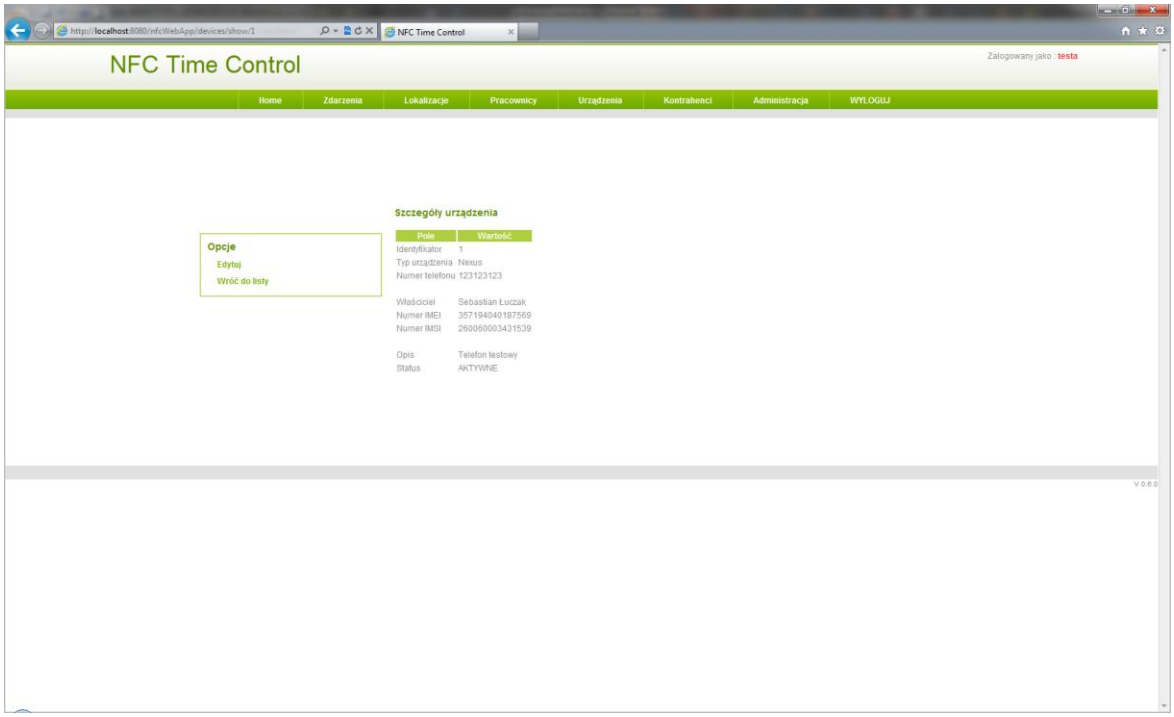


Rys. 5.14 Widok zakładki „Lokalizacje”

Po wybraniu pozycji „Urządzenia” z menu, zostaje przedstawiona lista telefonów, którymi posługują się pracownicy. Tak jak w poprzednich przypadkach, możliwe jest edytowanie jak i dodawanie nowych wpisów do systemu.

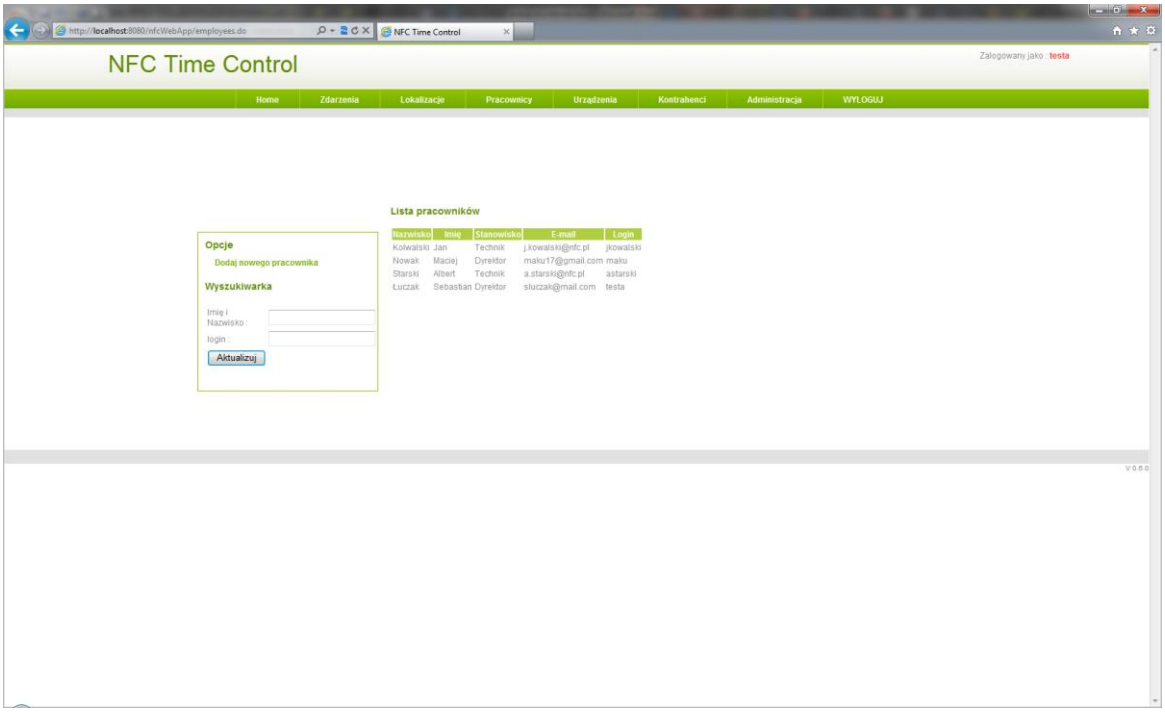


Rys. 5.15 Widok zakładki "Urządzenia"

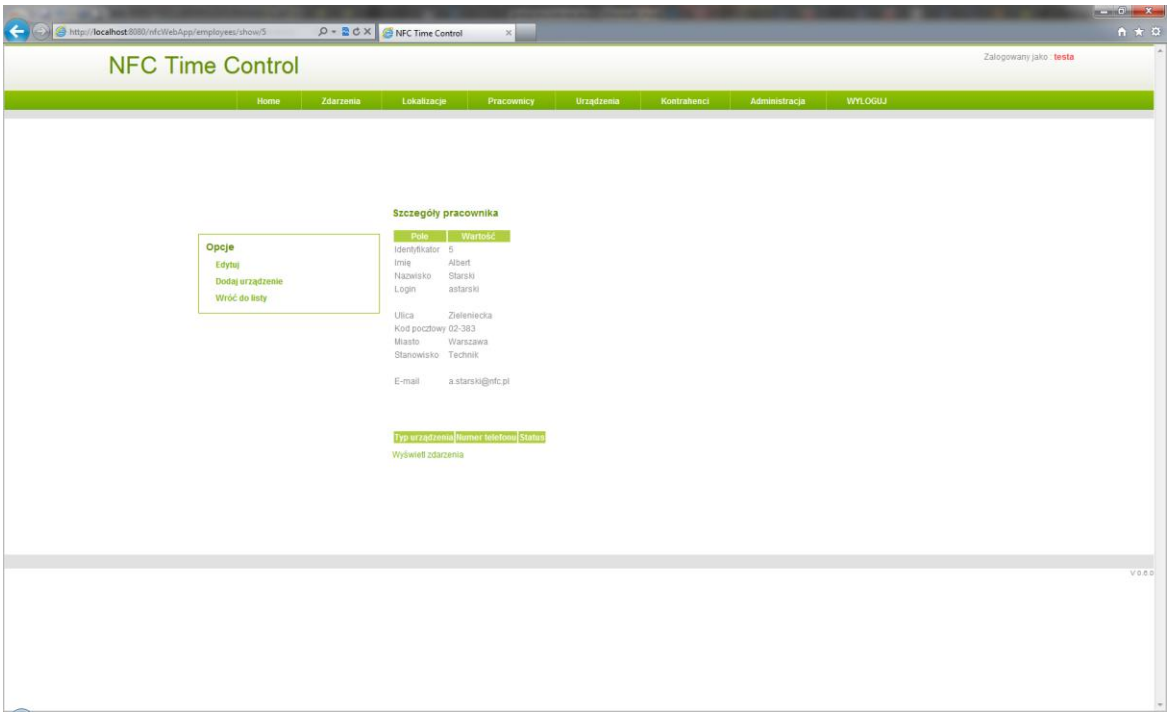


Rys. 5.16 Szczegółowy widok danych urządzenia

Kolejną pozycją w menu jest wpis „Pracownicy”. Jej wybranie powoduje wyświetlenie listy pracowników. Operator chcąc wyświetlić dane szczegółowe pracownika klika na odpowiednią pozycję na liście. Z poziomu zakładki wyświetlającej informacje szczegółowe możliwe jest przypisanie pracownikowi urządzenia bądź też w przypadku administratora nadanie dostępu do portalu. Wraz z szczegółowymi danymi pracownika wyświetlana jest lista przypisanych mu urządzeń, dodatkowo z przyciskiem umożliwiającym ich odłączenie.

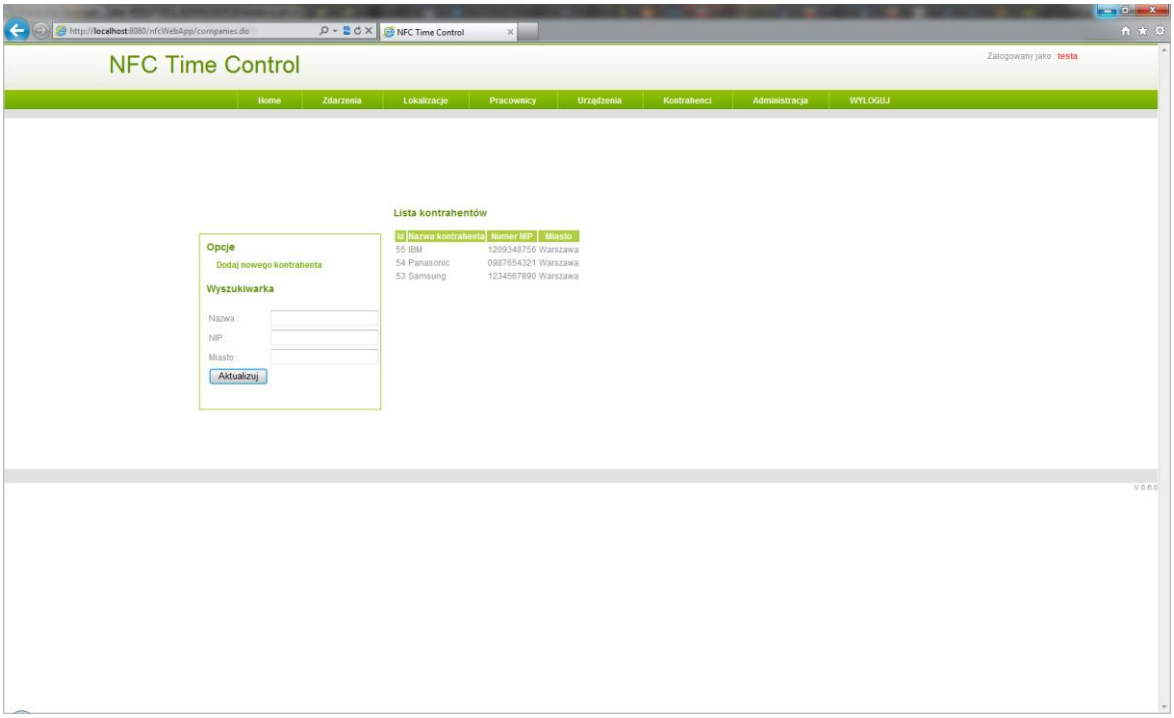


Rys. 5.17 Widok zakładki „Pracownicy”

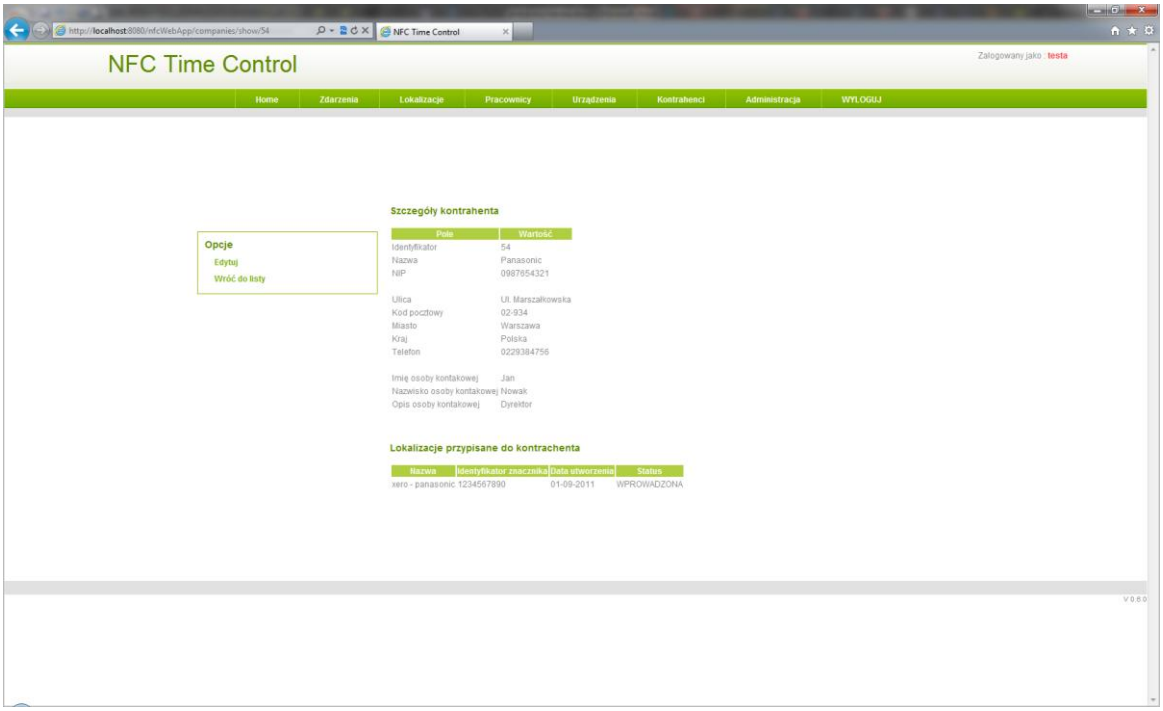


Rys. 5.18 Szczegółowy widok danych pracownika

Ostatnią opcją widoczną dla wszystkich użytkowników, jest zakładka „Kontrahenci”. Ma ona na celu przedstawienie listy oraz danych szczegółowych poszczególnych firm zarejestrowanych w systemie.

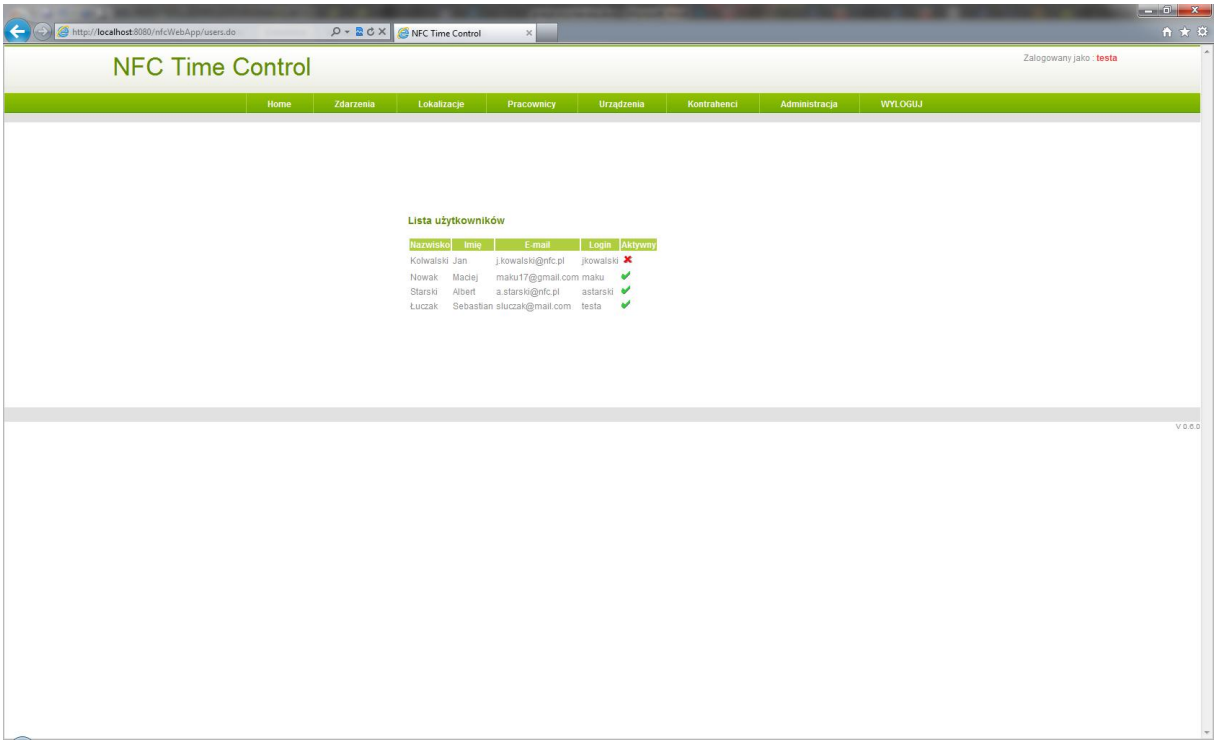


Rys. 5.19 Widok zakładki „Kontrahenci”

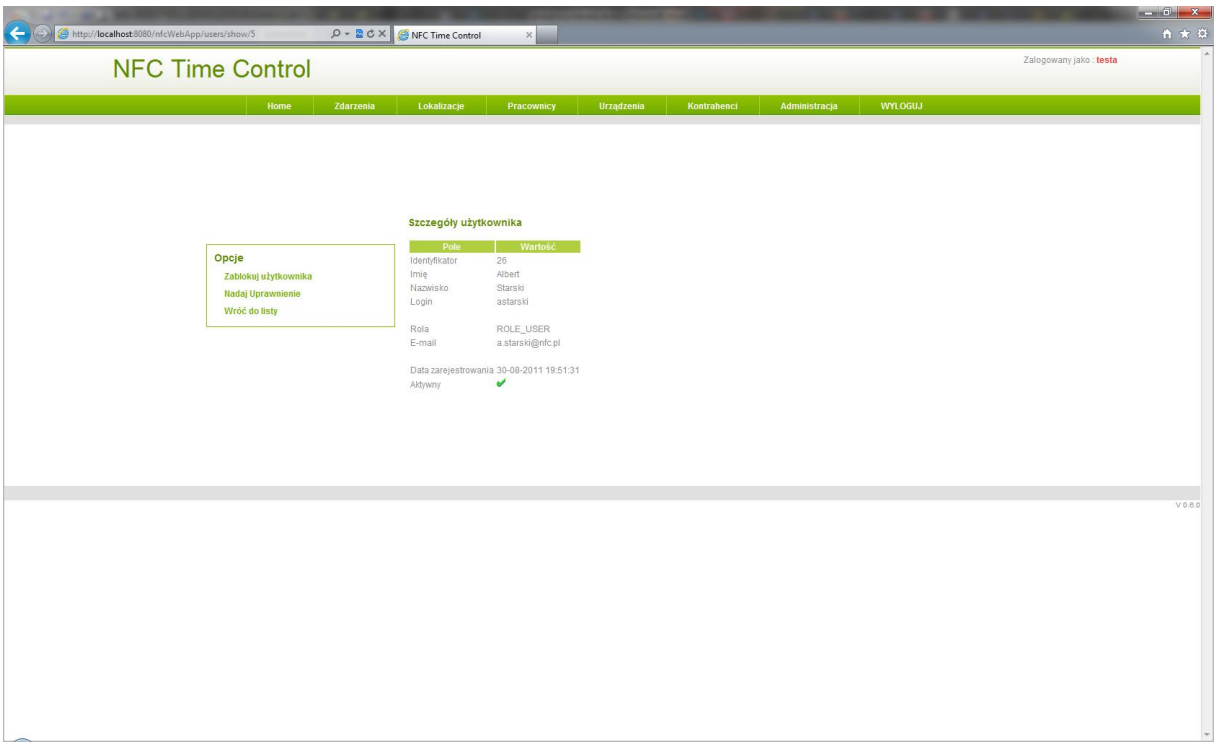


Rys. 5.20 Szczegółowy widok danych kontrahenta

Jak już wcześniej zostało wspomniane, użytkownik z uprawnieniami `ROLE_ADMIN` posiada dostęp do dodatkowej zakładki „Administracja” w której widnieje lista użytkowników oraz informację czy są oni aktywni. Dodatkowo administrator z tego poziomu może nadawać uprawnienia użytkownikom oraz ich blokować.



Rys. 5.21 Widok zakładki „Administracja”



Rys. 5.22 Szczegółowy widok danych użytkownika

6. TESTY FUNKCJONALNE

6.1. WSTĘP

Testy funkcjonalne systemu przebiegały w kilku scenariuszach, wliczając sytuacje poprawnego działania wszystkich modułów, braku dostępu do sieci, nieoczekiwanego wyłączenia serwera, błędnych danych wprowadzanych do formularzy itp.

We wszystkich przypadkach aplikacje stacjonarne uruchomione były na maszynie wirtualnej VirtualBox z zainstalowanym systemem operacyjnym Linux Ubuntu 10.10. Serwerem aplikacji był Apache Tomcat 6.0.32, a serwerem bazodanowym Oracle 10g XE.

Testy aplikacji mobilnych realizowano na telefonie Google Nexus S z fabrycznie zainstalowanym systemem operacyjnym Android Gingerbread 2.3.4. Na telefonie zostały zainstalowane trzy aplikacje[5] innych producentów, również obsługujące komunikację NFC. Wykonano to w celu sprawdzenia czy prawidłowo działają mechanizmy automatycznego wywoływania aplikacji klienckiej do obsługi znaczników zarejestrowanych poprzez aplikację administracyjną.

Podczas testów używano znaczników typu Mifare Ultralight w postaci miękkich naklejek.

Testy przeprowadzano na urządzeniach biurowych z obudowami z tworzywa sztucznego, głośnikach z obudową drewnianą oraz urządzeniach AGD o obudowach metalowych.

Na czas testów w systemie utworzeni zostali następujący użytkownicy o podanych uprawnieniach:

testa - `ROLE_ADMIN`

maku – `ROLE_SUPER_USER`

astarski – `ROLE_USER`

Poniżej przedstawione jest sprawozdanie z przebiegu testów z rozróżnieniem na poszczególne warianty.

[5] Zainstalowane aplikacje: NFC TagInfo, NXP TagWriter, NFC Tag Writer & Reader udostępniona na Android Market

6.2. WARIANT 1 – KONTROLA DOSTĘPU

Wykonano testy mające na celu weryfikację poprawności działania systemu kontroli dostępu.

6.2.1. Portal administracyjny

6.2.1.1. Przebieg

Przy pomocy każdego z wyżej wymienionych kont podjęto próby wyświetlenia każdej z dostępnych stron portalu administracyjnego, korzystając z łączy dostępnych po zalogowaniu.

Następnie przy ich pomocy, a także bez logowania do systemu, podjęto próby wyświetlenia każdej z dostępnych stron portalu administracyjnego, ręcznie wpisując do paska adresu przeglądarki strony dostępne tylko dla użytkowników `ROLE_SUPER_USER`, a następnie `ROLE_ADMIN`.

6.2.1.2. Rezultat

Zgodnie z oczekiwaniami aplikacja portalu udostępniała użytkownikom łącza zgodne z ich poziomem dostępu. Próby uzyskania dostępu do zabronionych obszarów serwisu poprzez ręczne wprowadzanie adresu zakończyły się fiaskiem. Za każdym razem aplikacja portalu zwracała błąd 403 Odmowa/Zakaz dostępu. W przypadku niezalogowania do systemu następowało automatyczne przekierowanie do strony logowania.

6.2.2. Kliencka aplikacja mobilna

6.2.2.1. Przebieg

Przy pomocy każdego z wyżej wymienionych kont podjęto próbę zalogowania się do mobilnej aplikacji klienckiej. Żadne z kont nie miało przypisanego telefonu z numerami IMEI i IMSI.

Dokonano ponowienia testów przypisując urządzenie do każdego konta z osobna.

6.2.2.2. Rezultat

Zgodnie z oczekiwaniami, przy braku przypisanego urządzenia, aplikacja kliencka informowała o braku dostępu i nie pozwalała na korzystanie z jej funkcji.

Po ponowieniu próby z poprawnie przypisanym urządzeniem, każdy z użytkowników otrzymał dostęp do funkcji aplikacji.

6.2.3. Administracyjna aplikacja mobilna

6.2.3.1. Przebieg

Przy pomocy każdego z wyżej wymienionych kont podjęto próbę zalogowania się do mobilnej aplikacji administracyjnej. Żadne z kont nie miało przypisanego telefonu z numerami IMEI i IMSI.

Dokonano ponowienia testów przypisując urządzenie do każdego konta z osobna.

6.2.3.2. Rezultat

Zgodnie z oczekiwaniami, przy braku przypisanego urządzenia, aplikacja kliencka informowała o braku dostępu i nie pozwalała na korzystanie z jej funkcji.

Po ponowieniu próby z poprawnie przypisanym urządzeniem, użytkownik astarski (ROLE_USER) nie został dopuszczony do aplikacji, z kolei użytkownicy testa (ROLE_ADMIN) i maku (ROLE_SUPER_USER) uzyskali do niej dostęp.

6.2.4. Podsumowanie

Test funkcjonalny kontroli dostępu w systemie został przeprowadzony z wynikiem pozytywnym.

6.3. WARIANT 2 – UŻYWANIE APLIKACJI MOBILNYCH W OBRĘBIE SIECI UMTS LUB WIFI

Przeprowadzono testy działania aplikacji mobilnych w obrębie sieci UMTS i WIFI. Scenariusz zakładał korzystanie z kont uprawnionych do użytkowania obu aplikacji i poprawne działanie usługi sieciowej przez cały czas trwania testu.

6.3.1. Aplikacja administracyjna

6.3.1.1. Przebieg

Zalogowano do aplikacji administracyjnej przy użyciu konta testa. Zostały wprowadzone dane lokalizacji i kontrahenta niewidniejącego jeszcze w bazie. Zgodnie z informacją wyświetloną na ekranie, zbliżono telefon do niezarejestrowanego znacznika umieszczonego na obudowie drukarki z tworzywa sztucznego. Znacznik został zarejestrowany poprawnie, o czym poinformował stosowny komunikat, a sprawdzenie zawartości bazy danych potwierdziło prawidłową rejestrację lokalizacji i nowego kontrahenta.

Ponowiono próbę rejestracji znacznika przy tych samych danych lokalizacji i kontrahenta oraz po ich modyfikacji. System podwójnie zgłosił błąd spowodowany próbą rejestracji znacznika który widnieje już w bazie danych.

Wprowadzono dane nowej lokalizacji i kontrahenta zarejestrowanego już w bazie danych, a następnie przeprowadzono rejestrację nowego znacznika. Proces przebiegł poprawnie, wyświetlony został stosowny komunikat, a zawartość bazy danych potwierdziła powiązanie nowej lokalizacji z istniejącym już kontrahentem

Wszystkie trzy testy powtórzono umieszczając znacznik na metalowej powierzchni obudowy lodówki. Czytnik NFC nie był w stanie skomunikować się ze znacznikiem, z tego powodu rejestracja się nie powiodła.

6.3.1.2. Rezultat

System rejestracji znaczników funkcjonuje prawidłowo w przedstawionych warunkach. Jedyną sytuacją w której rejestracja nie przebiega poprawnie, jest umiejscowienie znacznika na metalowej powierzchni.

6.3.2. Aplikacja kliencka

6.3.2.1. Przebieg

Test przeprowadzono z użyciem trzech znaczników, z których dwa były wcześniej zarejestrowane w systemie i powiązane z lokalizacjami.

Zalogowano do aplikacji klienckiej przy użyciu konta astarski. Na ekranie został wyświetlony status „Oczekiwanie…”. Pozostawiając aplikację uruchomioną, zbliżono telefon do zarejestrowanego znacznika. Użytkownik został poinformowany o rozpoczęciu realizacji zlecenia, a na ekranie zostały wyświetlone jego szczegóły.

Poprzez zbliżenie telefonu do drugiego znacznika, podjęta została próba rozpoczęcia drugiego zlecenia choć pierwsze nie zostało jeszcze zakończone. Użytkownik został poinformowany o błędzie wraz z przedstawieniem prawidłowej przyczyny.

Zbliżając telefon do pierwszego znacznika i potwierdzając swoją decyzję w oknie dialogowym, zakończono zlecenie. Na ekranie aplikacji na powrót został wyświetlony status „Oczekiwanie…”.

Próba zbliżenia telefonu do niezarejestrowanego znacznika zakończyła się błędem i wyświetleniem na ekranie przyczyny („Znacznik nie zarejestrowany w bazie”).

Wszystkie testy ponowiono w wariancie w którym aplikacja nie jest włączana przez użytkownika tylko automatycznie, po odczytaniu znacznika.

Zbliżanie telefonu do znacznika powodowało wywołanie aplikacji klienckiej i taki sam rezultat działania jak w przypadku pierwszej serii testów.

Testy powtórzono umieszczając znaczniki na metalowej powierzchni obudowy lodówki. Czytnik NFC nie był w stanie skomunikować się ze znacznikiem, z tego powodu rejestracja aktywności pracownika nie przebiegała pomyślnie.

6.3.2.2. Rezultat

System rejestracji aktywności pracownika funkcjonuje prawidłowo w przedstawionych warunkach. Jedyną sytuacją w której aplikacja kliencka nie działa poprawnie jest umiejscowienie znacznika na metalowej powierzchni.

6.3.3. Podsumowanie

Testy wykazały prawidłowe działanie systemu w sytuacji w której znacznik nie jest umieszczony na metalowej powierzchni. Sprawdzona została obsługa błędów wynikających z pomyłki użytkownika przy realizacji rejestracji znaczników i własnej aktywności.

6.4. WARIANT 3 – SYMULACJA TYMCZASOWYCH AWARII USŁUGI SIECIOWEJ LUB UTRATY ZASIĘGU

Sprawdzono działanie aplikacji mobilnych w sytuacji awarii usługi sieciowej lub utraty zasięgu. Scenariusz zakładał korzystanie z kont uprawnionych do użytkowania obu aplikacji.

6.4.1. Aplikacja administracyjna

6.4.1.1. Przebieg testów

Tymczasowo wyłączono serwer usługi sieciowej. Przeprowadzono serię testów z wariantu 2.1. We wszystkich przypadkach próby komunikacji z usługą sieciową, użytkownik otrzymywał komunikat o błędzie połączenia z serwerem. Aplikacja powracała do ekranu wprowadzania danych rejestrowanego znacznika.

Testy ponowiono w sytuacji braku dostępu do sieci transmisji pakietów. Osiągnięty rezultat był taki sam.

6.4.1.2. Rezultat

Aplikacja administracyjna poprawnie informuje użytkownika o wystąpieniu błędu w komunikacji, zachowując przy tym stabilność. Zgodnie z założeniami brak dostępu do sieci transmisji pakietów lub serwera, uniemożliwia poprawne działanie systemu.

6.4.2. Aplikacja kliencka

6.4.2.1. Przebieg testów

Tymczasowo wyłączono serwer usługi sieciowej. Przeprowadzono serię testów z wariantu 2.2. We wszystkich przypadkach próby komunikacji z usługą sieciową, użytkownik otrzymywał komunikat o błędzie połączenia z serwerem. Aplikacja powracała do ekranu głównego.

Testy ponowiono w sytuacji braku dostępu do sieci transmisji pakietów. Osiągnięty rezultat był taki sam.

6.4.2.2. Rezultat

Aplikacja kliencka poprawnie informuje użytkownika o wystąpieniu błędu w komunikacji, zachowując przy tym stabilność. Zgodnie z założeniami brak dostępu do sieci transmisji pakietów lub serwera, uniemożliwia poprawne działanie systemu.

6.4.3. Podsumowanie

Testy działania aplikacji w warunkach braku dostępu do usługi sieciowej (wywołanego przez jej awarię lub brak dostępu do sieci) wykazały, że obie aplikacje funkcjonują w tej sytuacji stabilnie, jednak zgodnie z założeniami nie zapewniają prawidłowego działania systemu.

6.5. TESTY INTERFEJSU UŻYTKOWNIKA APLIKACJI MOBILNYCH

W testach interfejsu użytkownika uczestniczyło 10 osób nie związanych z realizacją pracy dyplomowej w wieku od 23 do 55 lat. Cztery spośród nich korzystają na co dzień ze stacjonarnych systemów kontroli czasu pracy. Jedna zawodowo zajmuje się koordynacją czasu pracy pracowników mobilnych. Jedna jest pracownikiem mobilnym.

Podczas testów stosowane były konta uprawnione do obsługi obu aplikacji.

6.5.1. Aplikacja administracyjna

6.5.1.1. Przebieg testów

Testy z udziałem każdej osoby przebiegały niezależnie od siebie. Tester miał za zadanie bez żadnych podpowiedzi co do obsługi aplikacji zarejestrować lokalizację.
Jedyne informacje jakie otrzymywała każda osoba to:

- Które urządzenie ma zarejestrować
- Jakie informacje są niezbędne do rejestracji
- Dane kontrahenta

6.5.1.2. Rezultat

Żaden tester nie zgłosił zastrzeżeń co do interfejsu użytkownika aplikacji. Niektóre osoby potrzebowały pomocy przy obsłudze dotykowej klawiatury ekranowej systemu Android Gingerbread podczas wprowadzania danych. Wszyscy testerzy zdołali pomyślnie zarejestrować lokalizację w systemie.

6.5.2. Aplikacja kliencka

6.5.2.1. Przebieg testów

Testy z udziałem każdej osoby przebiegały niezależnie od siebie. Tester miał za zadanie bez żadnych podpowiedzi co do obsługi aplikacji zgłosić rozpoczęcie zlecenia i jego zakończenie.
Jedyne informacje jakie otrzymywała każda osoba to:

- Które *urządzenie* ma serwisować
- W celu wywołania aplikacji należy zbliżyć telefon do naklejki na obudowie urządzenia

6.5.2.2. Rezultat

Żaden tester nie zgłosił zastrzeżeń co do interfejsu użytkownika i nie potrzebował pomocy przy obsłudze aplikacji. Wszyscy testerzy pomyślnie zarejestrowali swoją aktywność w miejscu realizacji zlecenia.

6.5.3. Podsumowanie

Test interfejsu przebiegł pomyślnie. Użytkownicy nie zaznajomieni wcześniej z systemem nie mieli problemu z jego obsługą. Jedyne zgłoszone problemy związane były z obsługą dotykowej klawiatury ekranowej i związane były z brakiem wcześniej styczności danej osoby z telefonami o ekranach dotykowych.

Kompletne rozwiązanie spotkało się z dużym zainteresowaniem i entuzjazmem osoby zajmującej się zawodowo kontrolą czasu pracowników mobilnych.

6.6. WPŁYW APLIKACJI MOBILNYCH NA ZUŻYCIE BATERII TELEFONU

Podczas testów funkcjonalnych stwierdzono, ze ciągłe działanie aplikacji nie ma praktycznie żadnego wpływu na zużycie baterii. To samo tyczy się stale włączonego modułu NFC. Wbudowane w system operacyjny Android Gingerbread narzędzie monitorujące zużycie baterii wykazało, że jego poziom w odniesieniu do wspomnianych aplikacji utrzymuje się poniżej jednego procenta.

6.7. PODSUMOWANIE

System kontroli dostępu działa zgodnie z oczekiwaniami. Użytkownicy o zadanym poziomie uprawnień mają dostęp tylko do obszarów im udostępnionych.

Wszystkie testy funkcjonalne przeprowadzone w warunkach pełnej sprawności systemu i dostępu do sieci transmisji danych przebiegły pomyślnie.

W sytuacji utraty dostępu do sieci pakietowej, aplikacje mobilne nie są w stanie transmitować danych do serwera, co uniemożliwia ich poprawne działanie. Użytkownik jest jednak informowany o zaistniałej sytuacji zgodnie z założeniami.

W przypadku awarii serwerowej usługi sieciowej aplikacje mobilne nie są w stanie działać poprawnie. Użytkownik jest jednak informowany o przyczynie niesprawności zgodnie z założeniami.

Komunikacja bezstykowa NFC przy użyciu znaczników Mifare Ultralight nie działa prawidłowo gdy znacznik jest umieszczony na powierzchniach metalowych. Powierzchnia taka w wyniku odbić zakłóca sygnał radiowy, przez co ustanowienie prawidłowego połączenia nie jest możliwe.

W opinii testerów system jest intuicyjny i prosty w użytkowaniu, a jego obsługa nie absorbuje uwagi użytkownika.

Zarówno aplikacje mobilne przy normalnym użytkowaniu[6] jak i stałe działanie modułu komunikacji NFC powoduje zużycie energii baterii telefonu na poziomie poniżej 1% sumarycznego zużycia.

[6] Do trzydziestu zgłoszeń przy jednym naładowaniu baterii

7. PODSUMOWANIE

7.1. WADY I ZALETY ROZWIĄZANIA

Zaproponowane rozwiązanie jest odwróceniem klasycznego modelu kontroli czasu pracy. Dzięki temu możliwe jest zastosowanie go również w przypadku pracowników mobilnych.

Niewątpliwą zaletą zeń płynącą jest niski koszt implementacji, wdrożenia i utrzymania systemu. Efekt ten jest osiągany poprzez wykorzystanie telefonu jako czytnik i tanich znaczników typu Mifare Ultralight. Oparcie transmisji danych o sieci pakietowe eliminuje potrzebę budowy dodatkowej infrastruktury na potrzeby identyfikacji znacznika.

Dzięki przyłożeniu dużej wagi do interfejsu użytkownika, korzystanie z systemu jest intuicyjne i nie absorbujące pracownika, co wykazały testy.

Do zalet należy również zaliczyć brak potrzeby wyposażania pracownika w dodatkowe urządzenia, gdyż funkcję czytnika pełni posiadany przez niego telefon. Wartym zauważenia jest fakt, iż stałe działanie modułu NFC i aplikacji praktycznie nie wpływa na zużycie baterii.

Kolejną zaletą jest łatwość precyzyjnej kontroli czasu jaki spędził pracownik przy świadczonej usłudze, co dotychczas praktycznie nie było możliwe. Jest to tym bardziej istotne dla pracodawców którzy rozliczają się z pracownikami przy pomocy stawek godzinowych.

Żadne rozwiązanie nie jest jednak pozbawione wad. Do prawidłowego działania systemu niezbędne jest stałe połączenie z dowolną siecią umożliwiającą transmisję pakietów. Może to być bardzo utrudnione w przypadku lokalizacji położonych z dala od aglomeracji miejskich gdzie sieć telefonii komórkowej ma słabszy zasięg lub nie została rozwinięta.

Podczas testów okazało się również, że urządzenia o metalowych obudowach bardzo zakłócają pracę anten czytnika znajdującego się w telefonie i znacznika. Z tego powodu podczas rejestracji takich urządzeń niezbędne jest dobranie miejsca na obudowie skonstruowanego z innego materiału.

7.2. PERSPEKTYWY ROZWOJU

Dzięki konstrukcji przyjętej podczas budowy systemu, możliwe jest swobodne rozszerzanie jego funkcji o nowe moduły.

Przykładem możliwych propozycji rozwoju są:

- Moduł raportowy – ułatwiający pracodawcy generację raportów na temat czasu reakcji i realizacji powierzonych pracownikom zadań
- Rozbudowanie istniejącego modułu automatycznego powiadamiania pracowników o utworzeniu nowych zdarzeń – istnieje możliwość usprawnienia systemu rozsyłania wiadomości elektronicznych oraz wprowadzenia możliwości informowania użytkowników poprzez wiadomości SMS
- Konfigurowalne widoki statystyk i obecnego stanu pracowników – przydatnym rozszerzeniem byłaby możliwość graficznej wizualizacji zajętości i lokalizacji pracowników, a także możliwość dostosowania metody prezentacji do potrzeb użytkownika
- Wzbogacenie interfejsu użytkownika w telefonie komórkowym umożliwiające zarządzanie zdarzeniami nie tylko z poziomu portalu – obecnie w celu edycji bądź zamknięcia zdarzenia, użytkownik telefonu musi zalogować się na portal administracyjny
- Zarządzanie dostępem do obszarów – obecny system można zmodyfikować tak, aby umożliwiał kontrolę nad dostępem pracowników do określonych sekcji budynków, wymagałoby to jednak stworzenia dodatkowych układów elektronicznych.
- Użycie znaczników typu Label-On-Metal w celu umożliwienia poprawnej pracy systemu przy znakowaniu obiektów których obudowy są w całości metalowe.
- Stworzenie modułu dostarczającego użytkownikowi aplikacji klienckiej dostępu do informacji na temat historii napraw danego urządzenia lub listy części zamiennych dla danego typu urządzenia.

7.3. WNIOSKI

Przedstawiona praca dowodzi, że możliwe jest zbudowanie rozproszonego systemu kontroli czasu pracy dla pracowników mobilnych z wykorzystaniem nowych technologii, ze szczególnym uwzględnieniem Near Field Communication. Jest również przykładem na to, że technologia NFC, która tworzona jest głównie z myślą o zbliżeniowych formach płatności i zastosowaniach marketingowych, może zostać wykorzystana również w innych dziedzinach życia.

Cel zrealizowania systemu ułatwiającego nadzór nad pracownikami mobilnymi poprzez umożliwienie rzetelnego rozliczenia ich czasu pracy został osiągnięty. Łączy on w sobie funkcjonalność i atrakcyjność w sensie ekonomicznym, jednocześnie zapewniając dotychczas niedostępne możliwości kontroli czasu pracy. Zastosowane technologie pozwoliły osiągnąć bardzo niski koszt wdrożenia i utrzymania, dzięki czemu system może znaleźć zastosowanie w małych i średnich przedsiębiorstwach których profil zakłada zatrudnienie pracowników realizujących zadania w terenie. Proponowane rozwiązanie pozwala takim firmom efektywniej wykorzystywać dostępne środki i działa dyscyplinująco na pracowników, co ostatecznie zwiększyć może wydajność ich pracy. Jednocześnie, dzięki aplikacji portalu administracyjnego, klarowna prezentacja danych odnoszących się do realnego czasu poświęconego na dane zadanie, umożliwia bardziej precyzyjne ustalenie cen usług, adekwatnie do ponoszonych przez pracodawcę kosztów.

Co prawda do działania systemu nie jest angażowana lokalizacja satelitarna GPS, jednak powiązanie znaczników z fizycznymi lokalizacjami urządzeń, pozwala stworzyć mapę aktywności zatrudnionych osób w oparciu o adresy lokalizacji. Działanie takie pozwala monitorować obecną lokalizację pracowników nie angażując dodatkowych urządzeń. W konsekwencji daje możliwość lepszego zarządzania doborem zadań tak, aby minimalizować dystanse jakie każdy pracownik musi pokonać pomiędzy punktami realizacji kolejnych zleceń. W efekcie pracodawca uzyskuje kolejne oszczędności opierające się na kosztach transportu i poprawia jakość obsługi klienta, zapewniając krótszy czas oczekiwania na usługę.

Jak wykazały testy, interfejs aplikacji zapewnia niemal całkowitą transparentność systemu dla użytkownika i jego użytkowanie nie przeszkadza w realizacji obowiązków.

Wykorzystanie telefonu wyposażonego w moduł NFC w roli czytnika zbliżeniowego, pozwoliło na odwrócenie tradycyjnego modelu kontrolowania czasu pracy w oparciu o czytniki stacjonarne. Dzięki temu nie ma potrzeby wyposażania pracownika w dodatkowe urządzenia komunikujące się z systemem.

Zastąpienie tanimi, pasywnymi znacznikami Mifare Ultralight kosztownych punktów kontroli, które są zazwyczaj wymagającymi zasilania urządzeniami aktywnymi, zwiększyło niezawodność systemu przy jednoczesnej redukcji jego kosztów.

Oparcie komunikacji pomiędzy telefonem a serwerem na sieci Internet zapewniło ogólnodostępność i elastyczność stworzonego rozwiązania, a także możliwość jego rozproszenia. Ponieważ rozmiar danych transmitowanych pomiędzy serwerem a aplikacją mobilną jest bardzo niewielki [7], odległość pracownika od serwera praktycznie nie wpływa na czas obsługi zgłoszenia.

Użycie łatwo rozszerzalnych framework'ów do budowy portalu administracyjnego spowodowało, że aplikacja ta może być w prosty sposób rozbudowywana o kolejne funkcje w zależności od zapotrzebowania.

System jest jednak wrażliwy na dwie niesprzyjające sytuacje. Pierwszą z nich jest znakowanie urządzeń o metalowych obudowach. Komunikacja radiowa czytnika NFC ze znacznikiem ulega wtedy zakłóceniu i zgłoszenie zdarzenia nie jest możliwe. Drugą jest fakt iż poprawne działanie systemu jest uzależnione od dostępu użytkownika do sieci transmisji danych pakietowych, przez co staje się niezdatny do użycia w obszarach na których nie jest ona obecna.

Świadomość tych problemów pozwala na stwierdzenie iż system powinien być implementowany na terenie aglomeracji miejskich, a proces znakowania urządzeń wymaga od użytkownika aplikacji administracyjnej wyszukania obszaru obudowy urządzenia, pozbawionego elementów metalowych.

Jak wskazują analizy rynkowe zrealizowane przez firmę Frost & Sullivan, barierą która ogranicza zastosowanie systemów opartych o NFC jest dostępność i upowszechnienie telefonów wspierających tę technologię. Według prognoz przeszkody te zostaną wyeliminowane do roku 2015, w którym to ponad 53% rynku urządzeń typu smartphone wyposażonych będzie w moduły komunikacji bezstykowej [21]. Z tego powodu w nadchodzących latach opracowany system może znaleźć zastosowanie komercyjne.

[7] Zależnie od zawartości, około 1,5 KB

8. BIBLIOGRAFIA

[1] **StrongLink.** RFID Label SLB01. *StrongLink.* [Online] Sierpień 2011. [Zacytowano: 28 08 2011.] http://www.stronglink.cn/english/slb01.htm.

[2] **NFC Forum.** NFC Forum. *About the Forum.* [Online] 2011. [Zacytowano: 28 Sierpień 2011.] http://www.nfc-forum.org/aboutus/.

[3] *Near Field Communication (NFC) Technology and Measurements White Paper.* **Rohde & Shwarz.** München : Rohde & Shwarz, 2011.

[4] *Near Field Communication in the real world – part II.* **Innovision Research & Technology plc.** Cirencester : Innovision Research & Technology plc, 2006.

[5] **NXP Semiconductors.** *Mifare Ultralight Features and Hints.* Hopewell : NXP Semiconductors, 2006. strony 11-12.

[6] —. MF0ICU1. *MIFARE Ultralight contactless single-trip ticket IC.* Gratkorn : NXP Semiconductors, 2010.

[7] **Gartner.** *Worldwide Smartphone Sales to End Users by Operating System in 2Q11.* Stamford : Gartner, 2011.

[8] **Hashimi, Sayed, Komatineni, Satya i MacLean, Dave.** *Android 2 Tworzenie aplikacji.* [tłum.] Krzysztof Sawka. Gliwice : Helion, 2010.

[9] **Mednieks, Zigurd, i inni.** *Programming Android.* Sebastopol : O'Reilly Media, 2011. strony 21-26, 142-154, 396-404.

[10] **Google Inc.** Android 2.3.3 Platform. *Android Developers.* [Online] Google, Luty 2011. [Zacytowano: 28 Sierpień 2011.] http://developer.android.com/sdk/android-2.3.3.html.

[11] —. Near Field Communication. *Android Developers.* [Online] 12 Czerwiec 2011. [Zacytowano: 28 Sierpień 2011.] http://developer.android.com/guide/topics/nfc/index.html.

[12] **SpringSource Community.** Reference Documentation. [Online] 01 09 2011. http://static.springsource.org/spring/docs/3.1.0.M2/spring-framework-reference/pdf/spring-framework-reference.pdf.

[13] **Naci, Dai, Mandel, Lawrence i Rayman, Arthur.** *Eclipse Web Tools Platform – Tworzenie aplikacji WWW w jezyku JAVA.* Gliwice : Helion, 2008.

[14] **Mullins, Craig S.** The Database Report – July 2011. *THE DATA ADMINISTRATION NEWSLETTER.* [Online] 1 Lipiec 2011. [Zacytowano: 28 08 2011.] http://www.tdan.com/view-featured-columns/15299.

[15] **Crockford, Douglas.** RFC 4627. *The application/json Media Type for JavaScript Object Notation.* brak miejsca : Network Working Group, 2006.

[16] **Singh, Inderjeet i Leitch, Joel.** Gson User Guide. *gson.* [Online] Google, 13 Lipiec 2011. [Zacytowano: 25 Lipiec 2011.] https://sites.google.com/site/gson/gson-user-guide.

[17] **Collins-Sussman, Ben, W. Fitzpatrick, Brian i Pilato, C. Michael.** *Version Control with Subversion.* Sebastopol CA : O'Reilly Media, 2011.

[18] **Google Inc.** Reference. *Android Developers.* [Online] Google, 24 Sierpień 2011. [Zacytowano: 28 Sierpień 2011.] http://developer.android.com/reference/android/package-summary.html.

[19] **Codd, Edgar Frank.** A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM.* 1970.

[20] **NFC Forum.** T2TOP 1.1. *Type 2 Tag Operation Specification.* Wakefield : NFC Forum, 2011. strony 3-6.

[21] **Frost & Sullivan.** Frost & Sullivan. *Promised Market for NFC Effectively Commences in 2011 with Commercial Roll out within All Verticals.* [Online] 31 styczeń 2011. [Zacytowano: 04 09 2011.] http://www.frost.com/prod/servlet/press-release.pag?docid=223107191.

[22] **Dave, Minter i Linwood, Jeff.** *Hibernate Od Nowicjusza do Profesjonalisty.* brak miejsca : Power Net, 2007.

[23] **Jason, Price.** *Oracle Database 11g i SQL Programowanie.* Gliwice : Helion, 2009.

[24] **Marty, Hall, Brown, Larry i Chaikin, Yaakov.** *Java Servlets I JavaServer Pages .* Gliwice : Helion, 2009. Tom II.

[25] **GlassFish Community.** Jersey 1.8 User Guide. [Online] 01 09 2011. http://jersey.java.net/nonav/documentation/latest/index.html.
[26] **SpringSource Community.** Spring Security Reference Documentation. [Online] 01 09 2011. http://static.springsource.org/spring-security/site/docs/3.0.x/reference/springsecurity.pdf.

9. ZAŁĄCZNIKI

9.1. DODATEK A

W dodatku A został przedstawiony dokładny opis tabel znajdujących się w bazie danych.

Tabela *users* zawiera dane o użytkownikach mających dostęp do systemu. Zawiera następujące pola :

- ID – unikalny identyfikator wpisu tabeli – klucz główny
- USER_NAME – nazwa użytkownika
- USER_PASS – hasło użytkownika – w tabeli jest przechowywany skrót hasła
- USER_CREATION_DATE – data zarejestrowania użytkownika
- ACTIVE – status użytkownika – 1 to aktywny, 0 to nieaktywny

Tabela *user_privileges* zawiera dane o uprawnieniach jakie posiada dany użytkownik :

- ID - unikalny identyfikator wpisu tabeli – klucz główny
- USER_ID – identyfikator z tabeli users identyfikujący użytkownika - klucz obcy
- ROLE – rola jaka została przypisana użytkownikowi.

Tabela *employee* zawiera szczegółowe dane na temat pracowników zarejestrowanych w systemie.

- ID - unikalny identyfikator wpisu tabeli – klucz główny
- NAME – imię pracownika
- LAST_NAME – nazwisko pracownika
- STREET – ulica
- CITY – miasto
- POST_CODE – kod pocztowy
- POSITION – stanowisko zajmowane przez pracownika
- USERS_ID - identyfikator z tabeli users nadający dostęp pracownikowi - klucz obcy
- EMAIL – adres e-mail pracownika

Tabela *devices* przechowuje informacje na temat urządzeń jakimi posługują się pracownicy :

- ID - unikalny identyfikator wpisu tabeli – klucz główny
- DEVICE_TYPE – typ urządzenia
- DEVICE_SN – numer seryjny urzadzenia w przypadku telefonu jest to numer IMEI
- EMPLOYEE_ID – identyfikator z tabeli employee – klucz obcy
- DEVICE_STATUS – status urządzenia
- PHONE_NUMBER – numer telefonu danego urzadzenia
- DESCRIPTION – opis urządzenia
- IMSI – numer IMSI karty sim przypisanej danemu urządzeniu

Tabela *events* przechowuje informacje na temat zdarzeń obsługiwanych przez pracowników :

- ID - unikalny identyfikator wpisu tabeli – klucz główny
- EVENT_TYPE – typ zdarzenia
- EVENT_CREATION_DATE – data utworzenia zdarzenia
- EVENT_START_DATE – data rozpoczęcia zdarzenia
- EVENT_FINISH_DATE – data zakończenia zdarzenia
- USER_ID – identyfikator z tabeli users reprezentujący użytkownika który obsługiwał zdarzenie – klucz obcy
- EVENT_COMMENTS – komentarz do zdarzenia
- STATUS – status zdarzenia
- TAG_ID – identyfikator znacznika powiązanego ze zdarzeniem
- CREATOR_ID - identyfikator z tabeli users reprezentujący użytkownika który zarejestrował zdarzenie – klucz obcy
- TOKEN - token
- EVENT_SYSTEM_START_DATE – data rozpoczęcia zdarzenia w systemie
- EVENT_SYSTEM_FINISH_DATE – data zakończenia zdarzenia w systemie
- LOCATION_ID - identyfikator z tabeli locations reprezentujący lokalizację na którą jest stworzone zdarzenie – klucz obcy

Tabela *location* przechowuje informacje na temat lokalizacji powiązanych z identyfikatorami znaczników :

- ID - unikalny identyfikator wpisu tabeli – klucz główny
- STREET - ulica
- CITY - miasto
- POST_CODE – kod pocztowy
- DETAILS – opis lokalizacji
- TAG_ID – identyfikator znacznika – unikalny w skali tabeli
- NAME – nazwa lokalizacji
- CREATION_DATE – data zarejestrowania lokalizacji
- STATUS – status lokalizacji
- CREATOR_ID identyfikator z tabeli users reprezentujący użytkownika który zarejestrował lokalizacje – klucz obcy
- COMPANY_ID - identyfikator z tabeli company reprezentujący firmę która jest właścicielem lokalizacji – klucz obcy
- OBJECT_SERIAL_NUMBER – numer seryjny urządzenia powiązanego z lokalizacją

Tabela *company* przechowuje informacje na temat firm będących właścicielami lokalizacji :

- ID - unikalny identyfikator wpisu tabeli – klucz główny
- NAME – nazwa firmy
- STREET - ulica
- CITY - miasto
- POST_CODE – kod pocztowy
- COUNTRY - kraj
- PHONE_NUMBER – numer telefonu kontakowego
- CONTACT_PERSON_NAME –imię osoby kontaktowej
- CONTACT_PERSON_LAST_NAME – nazwisko osoby kontaktowej
- CONTACT_PERSON_DESCRIPTION – opis osoby kontaktowej
- NIP – numer NIP firmy
- EMAIL – adres e-mail firmy