POLITECHNIKA WARSZAWSKA

Wydział Elektroniki i Technik Informacyjnych

Instytut Telekomunikacji

**PRACA DYPLOMOWA**

**INŻYNIERSKA**

**Sebastian Łuczak, Maciej Nowak**

**TYTUŁ**

Kierujący pracą:

mgr inż. Marcin Golański

..........................................

ocena pracy

..........................................

data i podpis Przewodniczącego

Komisji Egzaminacyjnej

Warszawa, wrzesień 2011

Streszczenie w języku polskim

Streszczenie w języku angielskim

Życiorysy

**Curriculum Vitae**

**Dane personalne:**

|  |  |
| --- | --- |
| Imię i nazwisko: | Sebastian Łuczak |
| Adres zamieszkania: | J. S. Bacha 34a/22, 02-743 Warszawa |
| Telefon: | 0 793 116 772 |
| e-mail: | fester3000@gmail.com |
| Data urodzenia: | 21 czerwca 1988 |
| Stan cywilny: | wolny |

**Wykształcenie:**

|  |  |
| --- | --- |
| 2007 do dziś | Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, specjalność: Sieci i systemy telekomunikacyjne |
| 2005 – 2007 | XXVIII Liceum Ogólnokształcące im. Jana Kochanowskiego w Warszawie |
| **Doświadczenie zawodowe:** |  |
| 2008 do dziś | CD Projekt RED  Praca na stanowisku Quest Designer przy rozwijaniu fabuły, prototypowaniu i implementowaniu elementów projektu Wiedźmin 2: Zabójcy Królów w oparciu o autorski silnik |
| 2007 | Sylogic business intelligence  Budowa portalu www.bridgeagency.pl (web development), serwis sprzętu komputerowego |
| 2004 – 2007 | FOSS Polska  Sezonowa praca w dziale logistyki polegająca na przyjmowaniu i wydawaniu towarów, oraz pomocy w pracach biurowych |
| 2005 – 2006 | Przewodnictwo Szkolnemu Zarządowi Uczniowskiemu |

**Dodatkowe umiejętności:**

|  |  |
| --- | --- |
| języki: | – biegła znajomość języka angielskiego  – podstawowa znajomość języka rosyjskiego |
| inne: | – ukończony kurs efektywnej nauki i szybkiego czytania  – biegła umiejętność obsługi komputera klasy PC w  środowiskach Windows/Unix/Linux  – znajomość języków C, C++, C#, PHP, Java  – znajomość zintegrowanych środowisk programistycznych MS Visual Studio 2010, NetBeans 6.9.1, Eclipse SDK 3.6.2  – umiejętność pracy przy ekstremalnych rygorach czasowych  – umiejętność pracy w wielozadaniowych zespołach projektowych |

**Zainteresowania:**

|  |  |
| --- | --- |
|  | – rozrywka multimedialna, programowanie, rozwój  technologii IT, sprzęt komputerowy  – film, literatura piękna, śpiew, taniec, aktorstwo,  pisarstwo, historia Warszawy  – koszykówka, pływanie, żeglarstwo, turystyka rowerowa, squash |

**Spis Treści**

[1. Wstęp 9](#_Toc302336497)

[2. Cel i zakres pracy 10](#_Toc302336498)

[3. Wstęp teoretyczny 12](#_Toc302336499)

[3.1. Opis technologii 12](#_Toc302336500)

[3.1.1. Near Field Communication 12](#_Toc302336501)

[3.1.2. Android 15](#_Toc302336502)

[3.1.3. Google NEXUS S 16](#_Toc302336503)

[3.1.4. Znacznik NFC Forum Type 2 - Mifare Ultralight 14](#_Toc302336504)

[3.1.5. Hibernate 3 16](#_Toc302336505)

[3.1.6. Spring Framework 3 18](#_Toc302336506)

[3.1.7. Oracle 10g Express Edition 20](#_Toc302336507)

[3.1.8. Java 2 Enterprise Edition 20](#_Toc302336508)

[3.1.9. Apache Tomcat 21](#_Toc302336509)

[3.1.10. Architektura REST 21](#_Toc302336510)

[3.1.11. JSON - JavaScript Object Notation 22](#_Toc302336511)

[3.1.12. System kontroli wersji Subversion 1.6 23](#_Toc302336512)

[3.1.13. Eclipse SDK 3.6.2 + Java EE 23](#_Toc302336513)

[3.1.14. Eclipse SDK 3.6.2 + Android SDK 24](#_Toc302336514)

[4. Koncepcja Pracy 25](#_Toc302336515)

[4.1. Wstęp 25](#_Toc302336516)

[4.2. Założenia i budowa systemu 26](#_Toc302336517)

[5. Opis systemu 29](#_Toc302336518)

[5.1. Scenariusz użycia 29](#_Toc302336519)

[5.2. Baza danych 30](#_Toc302336520)

[5.2.1. Opis tabel 32](#_Toc302336521)

[5.2.2. Integralność danych 33](#_Toc302336522)

[5.2.3. Wyzwalacze i sekwencje 35](#_Toc302336523)

[5.3. Opis usługi sieciowej – Web Service 36](#_Toc302336524)

[5.3.1. Cel powstania 36](#_Toc302336525)

[5.3.2. Szczegóły implementacyjne 36](#_Toc302336526)

[5.3.3. Mechanizm uwierzytelnienia 39](#_Toc302336527)

[5.3.4. Mechanizm sprawdzania statusu zdarzenia 39](#_Toc302336528)

[5.3.5. Rejestracja nowego zdarzenia 40](#_Toc302336529)

[Wprowadzanie nowej lokalizacji i znacznika 42](#_Toc302336530)

[5.4. Opis aplikacji klienckiej na telefon komórkowy 44](#_Toc302336531)

[5.4.1. Cel powstania 44](#_Toc302336532)

[5.4.2. Założenia 44](#_Toc302336533)

[5.4.3. Komunikacja pomiędzy usługą internetową a użytkownikiem aplikacji 45](#_Toc302336534)

[5.4.4. Tryby uruchomienia 45](#_Toc302336535)

[5.4.5. Odczyt znacznika 46](#_Toc302336536)

[5.4.6. Dobór aplikacji do obsługi znacznika 47](#_Toc302336537)

[5.4.7. Wprowadzenie danych i komunikacja z usługą internetową 48](#_Toc302336538)

[5.4.8. Działanie aplikacji 48](#_Toc302336539)

[5.4.9. Wpływ na zużycie baterii 50](#_Toc302336540)

[5.5. Opis aplikacji administracyjnej na telefon komórkowy 51](#_Toc302336541)

[5.5.1. Cel powstania 51](#_Toc302336542)

[5.5.2. Główne założenia 51](#_Toc302336543)

[5.5.3. Działanie aplikacji 51](#_Toc302336544)

[5.6. Opis panelu administracyjnego 54](#_Toc302336545)

[5.6.1. Cel powstania 54](#_Toc302336546)

[5.6.2. Szczegóły implementacyjne 54](#_Toc302336547)

[5.6.3. Warstwa biznesowa 55](#_Toc302336548)

[5.6.4. Warstwa modelu danych 56](#_Toc302336549)

[5.6.5. Warstwa prezentacji 56](#_Toc302336550)

[5.6.6. Budowa panelu administracyjnego 57](#_Toc302336551)

[5.6.7. Role użytkowników oraz zabezpieczenie portalu 58](#_Toc302336552)

[5.6.8. Opis zakładek 58](#_Toc302336553)

[6. Testy 62](#_Toc302336554)

[7. Korzyści płynące z naszego rozwiązania 63](#_Toc302336555)

[8. Podsumowanie 64](#_Toc302336556)

[8.1. Perspektywy rozwoju 64](#_Toc302336557)

[8.2. Wnioski 65](#_Toc302336558)

[9. Bibliografia 66](#_Toc302336559)

[10. ZałącznikI 67](#_Toc302336560)

[10.1. Dodatek A 67](#_Toc302336561)

Wykaz używanych skrótów i pojęć

Pracownik mobilny

Intencja

akcja

Lokalizacja i urządzenie

Usługa sieciowa

Telefon, aparat, urządzenie mobilne

Unikatowy identyfikator

Znacznik

RFiD

NFC

API – Application Programming Interface – interfejsy programistyczne

NFC – Near Field Communucation

RFiD – Radio Frequency Identification

ORM – Object-Realtional Mapping – technika mapowania obiektów na encje bazy danych

POJO – Plain Old Java Object – obiekty Java

SQL – Structured Query Language – język używany do tworzenia zapytań do relacyjnych baz danych

HQL – Hibernate Query Language – język zapytań stworzony specjalnie dla Framework Hibernate

QBE – Query By Example – technika tworzenia zapytań w Framework Hibernate

JDBC – Java DataBase Connectiovity – interfejs łączący program Java z bazą danych

MVC – Model-View-Controlle – wzrorzec projektowy Model – Widok - Kontroler

JRE – Java Runtime Enviroment – środowisko uruchomieniowe programów Java

JVM – Java Virtual Machine – maszna wirtualna Java

J2EE – Java Enterprise Edition

EJB – Enterprise Java Beans – technologia programistyczna Java

JSP – Java Server Pages

XML – eXtesible Markup Language

HTML – HiperText Markup Language – język tworzenia stron WWW

WWW – World Wide Web

CSS - *Cascading Style Sheets -*

REST – Representational State Transfer

http – HiperText Transfer Protocol

URI – Uniform Resource Identifier

JSON – JavaScript Object Notation

SOAP – Simple Obcjec Access Protocol

JAX-RS – Java Api for Restfull Web Services

IDE – Integrated Development Enviroment

SDK – Software Development Kit

SHA – Secure Hash Algoritm

DAO – Data Access Object

JSTL – Java Standart Tag Library

# Wstęp

Na koniec

# Cel i zakres pracy

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

Dzięki użyciu infrastrukturzy telefonii 3G, oraz złożonego systemu monitorowania uzyskujemy pełną i pewną 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 wytwarzania 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 i kart zbliżeniowych, a także rozwiązaniem 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 na potrzeby 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 instalacja u kontrahentów, nie wymagająca żadnych narzędzi i ingerencji w otoczenie, a także zapewnienie transparentności i łatwości użytkowania dla pracowników.

Zarówno koncepcja jak i projekt aplikacji został wymyślony i zrealizowany wspólnie. Ze względu na modułowość zaprojektowanego przez nas rozwiązania, dokonaliśmy podziału 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 oraz portalu administracyjnego Maciej Nowak. Prace nad interfejsem portalu wykonane były wspólnie.

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

# Wstęp teoretyczny

Rozpoczynając projekt spędziliśmy kilka miesięcy na porównywaniu przeróżnych technologii i doborze optymalnych do realizacji naszego systemu. Pierwsza faza obfitowała w

liczne prototypy dzięki którym wyeliminowaliśmy błędne założenia i elementy nadmiarowe, które nie spełniały naszych oczekiwań, lub pogarszały efekt końcowy.

Pierwsze tygodnie implementacji stanowiły dowód, że gruntowne przygotowanie podłoża, ustalenie zbioru potrzebnych aplikacji i faza prototypowania były warte poświęconego czasu, gdyż dzięki temu, dysponując wszystkimi niezbędnymi narzędziami, mogliśmy zabrać się do konsekwentnego rozwoju projektu.

## Opis technologii

### Near Field Communication

Ponieważ od początku dążyliśmy do stworzenia systemu opartego na komunikacji zbliżeniowej, właśnie od niej rozpoczęliśmy rozpoznanie. Komunikacja zbliżeniowa zapewnić nam miała 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 rozważaliśmy właśnie tę technologię. Szybko jednak okazało się, że co prawda istnieją czytniki przenośne, ale są one drogie, relatywnie duże i nieporęczne. Nasze rozwiązanie miało pomagać pracodawcy i pracownikom, a nie utrudniać wykonywanie obowiązków i dodatkowo obciążać pracownika.

W dobie rozwoju funkcji oferowanych przez telefony komórkowe, wydało się nam dobrym pomysłem, żeby powiązać czytnik zbliżeniowy z takim właśnie urządzeniem. W ten sposób uzyskalibyśmy również łatwy i gotowy sposób transmisji danych oparty o istniejącą już infrastrukturę. Dodatkowo należy nadmienić, że grupa docelowa czyli pracownicy mobilni i tak jest zawsze wyposażona w telefony. Stwierdziliśmy, że dodanie jednej funkcji do już posiadanego urządzenia nie powinno być dla nich zbyt obciążające. Oczywiście zakładając, że interfejs użytkownika nie będzie wymagał od nich zbyt wielu dodatkowych działań.

Łącząc te dwa fakty, czyli komunikację w oparciu o standard RFiD i telefonię komórkową, odkryliśmy istnienie organizacji NFC Forum sprawującej pieczę nad rozwojem i propagowaniem nowego standardu komunikacji zbliżeniowej , a także prototypowe modele telefonów wyposażone w moduły komunikacji Near Field Communication.

Near Field Communication (NFC) stanowi grupę technologii krótkodystansowej radiowej komunikacji zbliżeniowej, stworzonych głównie z myślą o urządzeniach przenośnych takich jak telefony komórkowe i tablety. 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, a korzenie nowego standardu leżą właśnie w RFiD.

Komunikacja w NFC oparta jest o zjawisko indukcji magnetycznej powstałej pomiędzy dwiema antenami. Odpowiednia modulacja tego pola pozwala na transmisję danych. Dopuszczalne prędkości zawierają się w zakresie od 106 kbit/s do 424 kbis/s.

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 .

Ostatecznie, ponieważ oferowane przez ten standard możliwości pokrywały się z naszymi oczekiwaniami, zdecydowaliśmy się na wykorzystanie w naszym systemie komunikacji NFC do rejestrowania realizacji zleceń przez pracowników.

### Znacznik NFC Forum Type 2 - Mifare Ultralight

Jedną z kluczowych zalet prezentowanego systemu jest jego niska cena wdrożeniowa i eksploatacyjna. Między innymi z tego powodu zdecydowaliśmy się na wykorzystanie niezwykle tanich (0.44$ za sztukę przy zakupie 1000-4999 sztuk ) i prostych konstrukcyjnie znaczników NFC Forum Type 2 Mifare Ultralight w formie miękkiej nalepki, w której cały układ wprasowany jest w papier.

Znaczniki te cechują się pojemnością zaledwie 384 bitów dostępnych na dane użytkownika w trybie zapis/odczyt, wyposażone są w system unikania kolizji, oferują prędkość transmisji 106 kbit/s, a także dostarczają możliwość jednokrotnego i nieodwracalnego zablokowania zapisu na znaczniku. Wadą tego rozwiązania jest jednak niemożność zmiany kodów dostępu do odczytu sektorów, przez co wszystko co jest zapisane na znaczniku, może być odczytane przez czytnik zgodny ze standardem ISO/IEC 14443 .

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

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

****

Rysunek DODAĆ PODPIS

### Android

Kolejnym istotnym czynnikiem doboru technologii w której zrealizowana miała być część pracy zwana przez nas mobilną, była popularność danego rozwiązania na rynku konsumenckim tak, aby uzyskany system miał również zastosowanie pozanaukowe. Po zapoznaniu się z obecnym podziałem rynku okazało się, że 43,4% (Dane z Sierpnia 2011, Gartner ) 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, uniwersalności, jak i liczby zapowiedzianych modeli wyposażonych w tę technologię.

Konsekwencją popularności Androida jest duży dostęp do różnorodnych opracowań tematu 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.

W pierwszej fazie projektowania aplikacji zmuszeni byliśmy do uwzględnienia faktu iż Android Gingerbread w wersji 2.3 nie dostarczał interfejsu programistycznego pozwalającego na pełną obsługę NFC. Dopiero kolejne aktualizacje do systemu operacyjnego zniosły z telefonów sztuczne ograniczenia. W pierwszej wersji Gingerbread niemożliwe między innymi było zapisywanie informacji na znacznik NFC (tylko odczyt). Obsługiwana była mniejsza liczba dostępnych typów znaczników i nie istniała możliwość obsłużenia zbliżonego do urządzenia znacznika przez aplikację obecnie uruchomioną na pierwszym planie (system zawsze pytał się która aplikacja ma być wywołana dla danego znacznika) w sposób automatyczny.

Początkowy projekt aplikacji zakładał występowanie tych ograniczeń, szybko jednak okazało się, że w aktualizacji 2.3.3 (Luty 2011 [8] [8]) wspomniane funkcje zostały dostarczone, a dostępne wcześniej API zmodyfikowane i w wielu przypadkach zauważalnie poprawione [9] [9] [9].

Jak jednak uczy doświadczenie, każde rozszerzenie może przynieść rezultat odwrotny do oczekiwanego, dlatego też poddaliśmy pod ostrożną rozwagę poszerzenie funkcji realizowanych przez terminal mobilny.

Po pierwszych prototypach wykorzystujących nowe możliwości, okazało się, 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 . Dzięki tej funkcji zdołałem doprowadzić do sytuacji w której przy zbliżeniu do telefonu znacznika zarejestrowanego przez naszą aplikację administracyjną, aplikacja kliencka uruchamia się automatycznie.

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

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

### Google NEXUS S

Ponieważ pracownicy mobilni często wyposażeni są w zaawansowane urządzenia typu smartphone zapewnione przez pracodawców, stwierdziliśmy, że docelową platformą musi być właśnie rozwiązanie tego typu. W konsekwencji wyboru systemu operacyjnego Android i wymogu posiadania modułu NFC, dobór urządzenia na którym powstać miały aplikacje był już w istotnym zakresie zdeterminowany. Krótkie rozeznanie rynku zakończyło się stwierdzeniem faktu – jedynym ówcześnie dostępnym telefonem spełniającym te wymagania był Google Samsung Nexus S. Po oficjalnej aktualizacji oprogramowania do wersji Android 2.3.4, telefon spełnił wszystkie złożone w nim oczekiwania.

### Hibernate 3

Framework Hibernate w wersji 3, jest najbardziej popularnym rozwiązaniem programistycznym oferowanym na zasadach open-source, 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.

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 usuwa oraz 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).

W stworzonym przez nas 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.

### Spring Framework 3

Kolejnym Frameworkiem wykorzystywanym w naszej 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 [10] ułatwia i promuje dobre praktyki programistyczne.

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 niezależnych od siebie modułów, które mogą być stosowane niezależnie. Są nimi między innymi z pakietu Spring MVC oraz Spring Security, które wykorzystujemy w naszym 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 odpowiedzialnych za wyświetlanie danych. Wg wzorca warstwa biznesowa 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 . Przepływem pomiędzy tymi warstwami zarządza kontroler, który także otrzymuje dane wejściowe od użytkowników i zarządza generowaniem odpowiednich widoków.

Spring Security to inny pakiet projektu Spring używany w naszym systemie. Jego zadaniem jest zapewnienie bezpieczeństwa panelu administracyjnego. Dzięki udogodnieniom oferowanym przez ten pakiet, w łatwy sposób możemy 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.

Dzięki swojemu zaawansowaniu wspiera on wiele metod uwierzytelnienia, takich jak na przykład LDAP czy też protokół Kerberos. W naszym przypadku uwierzytelnianie użytkownika opiera się na sprawdzeniu jego danych z informacjami zapisanymi w bazie danych.

### Oracle 10g Express Edition

W naszym projekcie zdecydowaliśmy się na wykorzystanie systemu zarządzania bazą danych Oracle w wersji 10g Express Edition, stworzonego przez firmę Oracle Corporation. Edycja XE, jest to darmowa , lecz 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, mieliśmy szansę łatwego zbudowania schematu bazy danych, a także jego późniejszego zaimplementowania.

Wybraliśmy tę konkretną bazę danych firmy Oracle, ze względu na chęć stworzenia systemu opartego na profesjonalnych komponentach składowych. Jednym z jego atutów jest przenośność na różne platformy sprzętowe i systemowe. Według danych za rok 2011 Oracle Corporation posiada prawie 50% rynku baz danych na świecie . Dzięki swojej popularności nie ma żadnych problemów w integracji bazy danych z frameworkiem Hibernate.

### 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 np. Windows, Linux, Solaris. Jest to możliwe dzięki stworzeniu środowiska uruchomieniowego JRE (ang. Java Runtime Enviroment ), które składa się z wirtualnej maszyny 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 . W jej skład wchodzą serwery, komponenty EJB (ang. Enterprise JavaBeans), usługi Web Service, pliki JSP, XML oraz inne elementy ściśle związane z wytwarzaniem aplikacji WWW takie jak HTML i CSS.

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

### Apache Tomcat

Nierozłącznym elementem aplikacji WWW jest serwer. W naszym projekcie wykorzystujemy darmową, rozpowszechnianą na zasadzie open-source implementację 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 (ang. Java Server Pages). Jest on w pełni kompatybilny z wymaganiami aplikacji opartych na J2EE.

### 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 .

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 naszej 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.

### JSON - JavaScript Object Notation

Wybierając format danych zależało nam na prostocie, niezależności od platformy i łatwości wyszukiwania błędów. Ostatecznie zdecydowaliśmy się na 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 staje się idealnym narzędziem do transmisji uporządkowanych danych poprzez sieć .

JSON jest alternatywą dla powszechnie wykorzystywanego w Internecie języka XML (ang. eXtensible Markup Language), który wymaga znacznie większych nakładów programistycznych – dodatkowe klasy i obiekty – przy przetwarzaniu danych.

W naszym 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 [14].

Poniżej, prezentujemy przykładowy format danych, przesyłany w naszym systemie pomiędzy telefonem komórkowym a usługą sieciową.

//Instancja klasy CResponse zapisana w notacji JSON:

{

„STATUS” : „EVENT\_STATUS\_CREATED”,

„MESSAGE” : „przykładowa wiadomość”,

„TOKEN”: „2cabeae9a6c1febc74f2f8d9af6391104952445f”

}

### 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 naszego projektu zdecydowaliśmy się na darmowe, sprawdzone rozwiązanie w postaci repozytorium Subversion 1.6. Dzięki temu uzyskaliśmy łatwy sposób na 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[15], które do bezproblemowej i szybkiej pracy wymaga hostingu o dużej gwarantowanej przepustowości. Z tego powodu zdecydowaliśmy się na skorzystanie z usługi *code.google.com*, która prócz innych rozwiązań ułatwiających prowadzenie projektu, udostępnia również repozytorium Subversion w wersji 1.6.

### 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 wolnego oprogramowania. 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.

Przy rozwoju aplikacji stacjonarnych zdecydowałem się na wykorzystanie tego środowiska ze względu na możliwość wykorzystania zawartej w nim platformy Web Tools Platform. Projekt ten 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.

### 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) [8] [8], korzystanie z samego Source Development Kit dostarczonego przez Google jest bardzo uciążliwe. W celu usprawnienia pracy nad aplikacjami mobilnymi użyłem zintegrowanego środowiska programistycznego (IDE – Integrated Development Environment**)** Eclipse SDK 3.6.2 połączonego ze wspomnianym zestawem narzędzi Android SDK.

Dzięki temu połączeniu uzyskałem dostęp do złożonej aplikacji typu debugger, możliwości wglądu w wydruki konsoli systemu Android poprzez narzędzie LogCat, a praca z kodem źródłowym została znacznie przyspieszona dzięki narzędziom podpowiadania składni .

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

# Koncepcja Pracy

## Wstęp

W poniższym rozdziale postaramy się przybliżyć główne założenia oraz budowę wymyślonego i zaprojektowanego przez nas systemu.

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

W naszym rozwiązaniu dokonaliśmy odwrócenia 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 mówimy o stałej lokalizacji pracowników i ich dużej liczbie – biurach, serwisach 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 dostarczającą pożądane dane.

W scenariuszu przez nas proponowanym czytnikiem jest telefon, a zatem urządzenie w które w dzisiejszych czasach każdy pracownik mobilny jest wyposażony. Nie ma zatem 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 ). Ostatni element generujący dodatkowe koszty, czyli kablową infrastrukturę, zastępujemy z kolei wykorzystywaniem gotowej infrastruktury sieci komórkowej i jej technikach pakietowej transmisji danych. Dzięki temu drastycznie zmieniamy rozkład kosztów i istotnie je obniżamy, jednocześnie osiągając efekt niedostępny wcześniej, czyli możliwość precyzyjnego określenia czasu pracy pracowników rozproszonych poza siedzibą firmy.

CO Z OPRORGRAMOWANIEM??

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.

Oczywiście zauważalną wadą zaproponowanego rozwiązania, jest wrażliwość na dostępność usług transmisji danych. Niestety wprowadzenie opcji pracy w trybie offline mogłoby spowodować ogromną wrażliwość na przekłamanie danych docierających do bazy, a raczej celową modyfikację daty systemowej przez użytkowników. Co prawda baza przechowuje dla każdego wpisu zarówno znacznik czasowy terminala użytkownika jak i serwera obsługującego zdarzenie. Jednak zakładając działanie w trybie offline i kolejkowanie wysyłania danych do serwera, użytkownik mógłby celowo emulować przebywanie poza zasięgiem sieci tak, aby zamaskować opóźnione rozpoczęcie lub zakończenie realizacji zlecenia.

## 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 i poprzez wymianę w formacie JSON 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:

// to do obrazek

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. Jednocześnie jest to zabezpieczenie przed błędami o naturze ludzkiej.
* 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.
* Nie może być sytuacji, w której na ten sam znacznik 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 - przyjmujemy, że 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.

Wszystkie te założenia zostały zaimplementowane w kodzie i logika aplikacji pilnuje aby ich przestrzegać.

# Opis systemu

## Scenariusz użycia

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

Przypuśćmy sytuację w której firma X zajmuję 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 Mifare Ultralight 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 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.

## Baza danych

### Cel powstania

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

### 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 , opisał po raz pierwszy zasady grupowania danych oraz panujących pomiędzy nimi zależności.

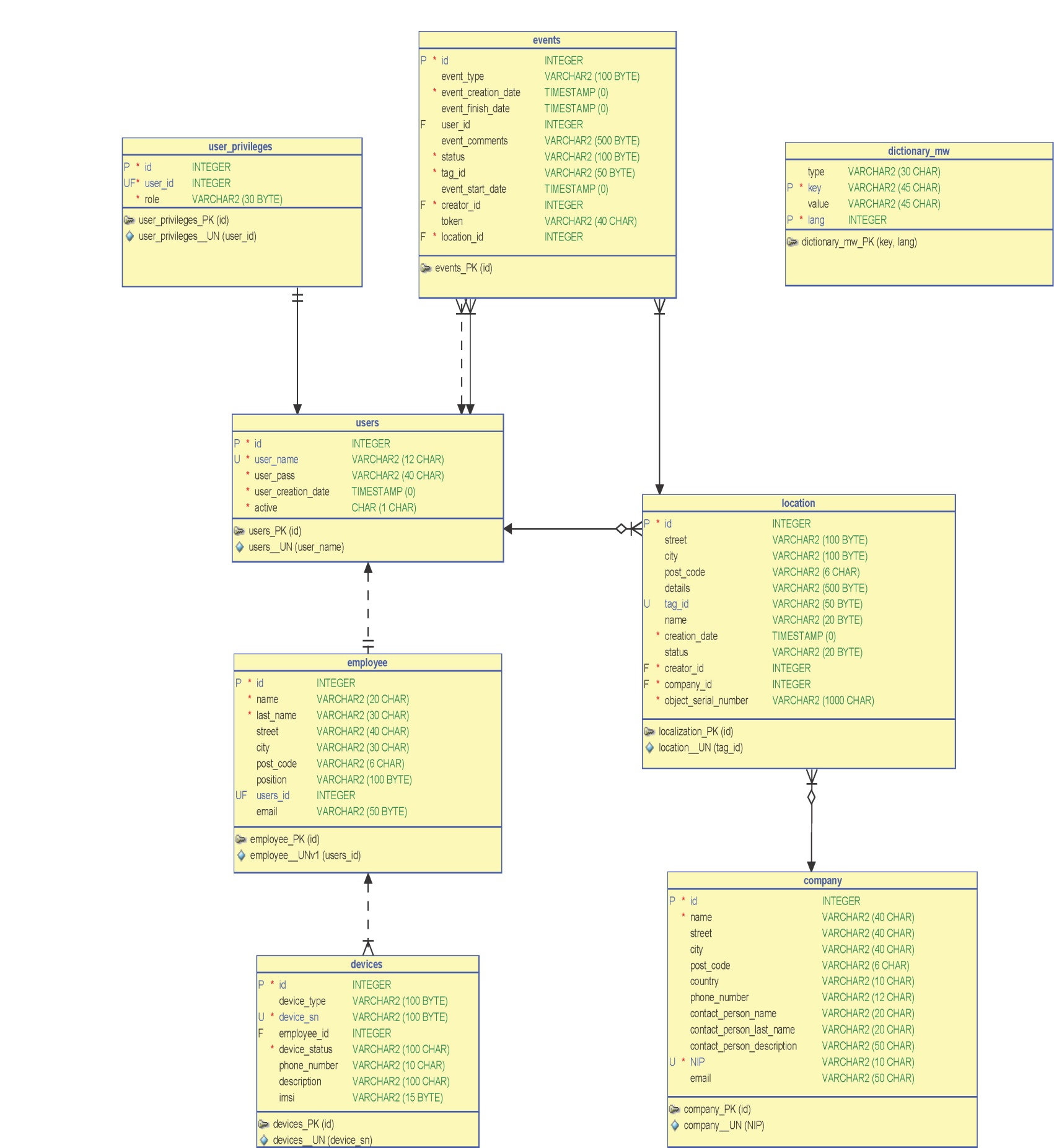
Model relacyjny zakłada grupowanie danych w relacje, które są reprezentowane za pomocą tabel. Są one zbiorem rekordów o ustalonej i identycznej strukturze wewnętrznej. Każda z nich posiada nazwę, nagłówek oraz zbiór atrybutów, zawartością relacji są z koleikrotki. Krotka jest jednoznacznie identyfikowana 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 (ang. Structured Query Language).

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

Na rysunku … został przedstawiony schemat użytej w systemie bazy danych.



*Rysunek …Schemat stworzonej bazy danych*

### Opis tabel

Poniżej przedstawiono skrótowy opis poszczególnych tabel z powyższego diagramu. Dokładny opis każdej z tabeli, ich 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 to czy użytkownik jest aktywny czy też nie.

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 ona identyfikator użytkownika, będący kluczem obcym pozwalający na określenie loginu 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 sa 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.

### Integralność danych

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

Za pomocą klauzuli *check,* założonej na tabelach *events*, *devices* oraz *locations* sprawdzam 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 …Lista używanych statusów obiektów*

Dodatkowo, stworzyłem mechanizm obejmujący tabelę *events*, niepozwalający na zamieszczenie w niej dwóch wpisów z tym samym TAG\_ID będących w statusie EVENT\_STATUS\_STARTED lub EVENT\_STATUS\_CREATED. Poniższy diagram pokazuje jakie sytuacje nie mogą mieć miejsca:

|  |  |  |
| --- | --- | --- |
| TAG\_ID | STATUS |  |
| Abc | EVENT\_STATUS\_CREATED | yes.gif |
| Abc | EVENT\_STATUS\_STARTED | yes.gif |
| Abc | EVENT\_STATUS\_FINISHED | yes.gif |
| Abc | EVENT\_STATUS\_FINISHED | yes.gif |
| Abc | EVENT\_STATUS\_CLOSED | yes.gif |
| Abc | EVENT\_STATUS\_CLOSED | yes.gif |
| Abc | EVENT\_STATUS\_CREATED | no.gif |
| Abc | EVENT\_STATUS\_STARTED | no.gif |

*Tabela … Przedstawienie mechanizmu integralności danych*

Poniżej zamieszczam 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 … Zapewnienie poprawności magazynowanych zdarzeń*

### Wyzwalacze i sekwencje

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

Poniżej przedstawiam 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 … Przykład wyzwalacza oraz sekwencji tabeli events*

## Opis usługi sieciowej – Web Service

### Cel powstania

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

### Szczegóły implementacyjne

Stworzona przeze mnie 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 zawierających się w module Entity 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. Widzimy na nim w jaki sposób nazwy pół, oraz ich typy są mapowane na nazwy kolumn z encji bazy danych. Kolejną 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 … 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 opiszę każdą z wyżej wymienionych wcześniej funkcji a także zaprezentuje przypadki wykorzystania ich w aplikacji. Zostaną także przedstawione algorytmy obsługujące nadchodzące zgłoszenia.

### 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 */nfcTimeControll/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.

### 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.

### Rejestracja nowego zdarzenia

Mechanizm rejestracji nowego zdarzenia oraz odnotowania zakończenia już istniejącego, jest podstawową funkcjonalnością systemu. Usługa sieciowa nasłuchuje na adresie */nfcTimeControl/service/reg* zgłoszenia przychodzące z aplikacji mobilnych.

Każde zgłoszenie przychodzące od telefonu komórkowego, zawiera dane pozwalające na uwierzytelnienie pracownika oraz dodatkowe informacje na temat rozpoczętego czy też zakończonego zdarzenia. Dzięki unikalnemu identyfikatorowi, znacznik który jednoznacznie określa lokalizację oraz dacie rozpoczęcia, 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 1.1



*Rys … Algorytm obsługujący zgłoszenie rejestracji zdarzenia*

Na podstawie powyższego schematu, możemy zobaczyć 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 zostało przedstawiony cykl życia nowo zarejestrowanego zdarzenia.



*Rys … 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 informację 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 naszym systemie stosowany jako unikatowy identyfikator kontrahenta.

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

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

Uproszczony mechanizm działania tej funkcjonalności przedstawia rysunek 1.2



*Rys … Algorytm rejestrowania nowej lokalizacji*

Ze schematu możemy 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 podpięta pod istniejącą już firmę. W przeciwnym wypadku system zarejestruje nową firmę, lokalizację, a następnie zwróci użytkownikowi odpowiedni status opisujący pomyślne zakończenie procesu.

## Opis aplikacji klienckiej na telefon komórkowy

### Cel powstania

Przeznaczeniem aplikacji klienckiej dla pracownika mobilnego jest pełnić funkcję intuicyjnego i prostego w obsłudze interfejsu do systemu monitorowania stanu realizacji zleceń.

### Założenia

Główne założenia które stanowiły podstawę przy projektowaniu tej części systemu to możliwie jak największa szybkość wywołania i działania aplikacji, oszczędność baterii, 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. Komunikacja z użytkownikiem w oparciu o graficzny interfejs
5. Duża szybkość reakcji na działania użytkownika
6. Komunikacja 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 (Representational State Transfer) opiera się o zapytania w formie JSON (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, idące jednak w parze z kompletnym wsparciem .

### 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.)

### 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.

#### 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 [8] [8] dane logowania i jeśli nie może ich znaleźć, oferuje użytkownikowi możliwość wprowadzenia ich bądź opuszczenie aplikacji. Ponieważ przy każdym odczytaniu znacznika dane zdarzenia transmitowane do serwera opatrzone są kompletem danych identyfikacyjnych (z racji przyjętej architektury aplikacji typu RESTful nie posługujemy się sesjami dla użytkowników tylko wymianą zapytań 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 prawdziwej 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.

#### 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.

### Odczyt znacznika

Standard NFC dla znaczników NFC Forum Type 2 zapewnia prędkość transmisji 106 kbit/s, a sam znacznik posiada zaledwie 384 bitów pamięci na dane użytkownika. Na drodze testów okazało się,że zapis jakichkolwiek danych na samym znaczniku mija się z celem. Przyczyna leży w niewielkiej pojemności pamięci wybranego przez nas znacznika i unikaniu rozpraszania danych. Przechowywanie kompletu informacji w bazie danych zapewnia ich integralność .

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] [8]. W przypadku naszej aplikacji przetwarzamy jedynie unikatowy identyfikator znacznika, który przesyłany jest do bazy danych. Odczyt wiadomości NDEF ma miejsce jedynie do konsoli w celach testowych.

### Dobór aplikacji do obsługi znacznika

W wersji Android 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 . W tym celu stworzono dwa systemy wywoływania aplikacji.

#### 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 [18] [18] [18]. Poniżej przedstawię pokrótce 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.

#### 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] [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.

### Wprowadzenie danych i komunikacja z usługą internetową

Weryfikacja autentyczności użytkownika, z powodu upatrywanych zagrożeń w postaci ludzkiej pomysłowości i złej woli, nie może przebiegać z użyciem samej nazwy użytkownika i hasła. Z tego powodu do grupy danych którymi musi się uwierzytelnić aplikacja kliencka, doszły numer IMEI (International Mobile Equipment Identity) oraz IMSI (International Mobile Subscriber Identity), dzięki czemu upewniamy się, że użytkownik o danym identyfikatorze zgłasza swoją aktywność przy pomocy swojego telefonu (numer IMEI) oraz swojej karty SIM (numer IMSI), co eliminuje zagrożenia płynące z wymiany telefonami i danymi autoryzacyjnymi pomiędzy użytkownikami systemu.

|  |  |
| --- | --- |
| Parametr | Opis |
| Identyfikator użytkownika | Przydzielony przez administratora systemu, unikatowy w skali systemu |
| Hasło | Spełniające wymogi bezpieczeństwa **WYMOGI** |
| 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. |

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.

Dodatkowo dane przesyłane do serwera są poddane działaniom kryptograficznym w postaci szyfrowania symetrycznego *AES 256bit*, dzięki czemu użytkownik nie może z poziomu innego terminala *udawać*, że jest użytkownikiem systemu.

### 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 godzina jego rozpoczęcia. Informację o stanie użytkownika aplikacja pozyskuje w procesie synchronizacji z usługą internetową tuż po uruchomieniu aplikacji.

W sytuacji chęci rozpoczęcia zlecenia, użytkownik musi jedynie zbliżyć telefon do znacznika przyklejonego na obudowie urządzenia. 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.



PODPIS

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 zakończy się fiaskiem, gdyż logika aplikacji serwerowej na to nie dopuści. 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 - Application Not Responding).

## Opis aplikacji administracyjnej na telefon komórkowy

### 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.

### Główne założenia

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

### 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.

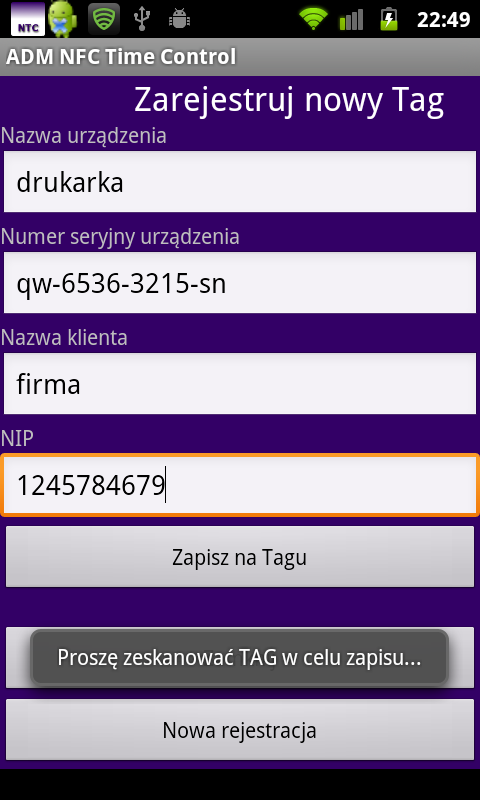
Ponieważ często może mieć miejsce rejestracja wielu urządzeń u jednego kontrahenta, pole zawierające NIP nie czyści się w jednej sesji działania aplikacji, tylko po jej wyłączeniu i ponownym włączeniu. Dzięki temu proces ten podlega kolejnemu uproszczeniu. Wybranie przycisku „Zapisz na znacznik” rozpoczyna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

Po pomyślnym przejściu walidacji pól, aplikacja zwraca się do użytkownika z prośbą o zbliżenie znacznika i korzystając z systemu Foreground dispatch [18] dostrojonego tak, aby reagował tylko na znaczniki typu Mifare Ultralight, uzyskuje najwyższy priorytet przy jego obsłudze.

W następnym kroku, jeśli użytkownik zbliży znacznik, aplikacja odczytuje jego unikatowy identyfikator, wysyła zapytanie do usługi internetowej i, jeśli operacja przebiegnie pomyślnie, zapisuje na znaczniku nazwę aplikacji oraz 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 zaznaczyłem 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 tworzyć (ale nie rejestrować!), postanowiłem zapisywać na znaczniku nazwę aplikacji przy użyciu własnego typu *text/vnd.timecontrol*.

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



PODPIS

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.

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.

## Opis panelu administracyjnego

### Cel powstania

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

### 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 naszym przypadku, zdecydowałem się na użycie serwera aplikacyjnego Apache Tomcat w wersji 6.0.32. Dzięki niemu mogłem stworzyć warstwę prezentacji opartą 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 który to został już opisany we wcześniejszej części pracy. Poniżej prezentuje opis struktury stworzonego przeze mnie panelu.

### 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.

### 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.

### Warstwa prezentacji

Warstwa prezentacji jest odpowiedzialna za poprawne wyświetlanie danych w ramach graficznego interfejsu użytkownika. Została ona wykonana za pomocą stron JSP (ang. Java Server Pages) wraz z użyciem biblioteki JSTL (ang. Java Standard Tag Library), 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 położyłem 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.

### 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… Schemat budowy panelu administracyjnego*

### 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 możemy 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.

### Opis zakładek

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

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.

Każdy wyświetlany na liście wynik jest hiperłączem do zakładki wyświetlającej szczegółowe dane o danej pozycji z możliwością ich edycji.

screen

Zakładka „Lokalizacje” ma użytkownikowi przedstawia 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.

screen

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.

screen

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.

screen

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

screen

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.

//screen

# Testy

York

Lodówka

Miasto i 3G

Wifi

Różne scenariusze

### Wpływ na zużycie baterii

Podczas testów funkcjonalnych stwierdzono, ze ciągłe działanie aplikacji nie ma praktycznie żadnego wpływu na zużycie baterii. To samo tyczy się stale włączonego modułu NFC (zużycie poniżej 1% baterii).

# Korzyści płynące z naszego rozwiązania

Zaproponowane przez nas rozwiązanie jest odwróceniem klasycznego modelu kontroli czasu pracy, między innymi dzięki temu możliwe jest zastosowanie go również w przypadku pracowników mobilnych.

Niewątpliwą zaletą płynącą z naszego rozwiązania jest niski koszt implementacji, wdrożenia i utrzymania systemu. Efekt ten uzyskujemy 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 uwagi 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.

# Podsumowanie

## Perspektywy rozwoju

Dzięki konstrukcji przyjętej podczas budowy naszego 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 – chcielibyśmy usprawnić system rozsyłania wiadomości elektronicznych oraz wprowadzić możliwość informowania 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.

## Wnioski

Na koniec

# Bibliografia

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

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

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

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

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

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

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

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

[9] —. 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.

[10]

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

[12] **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.

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

[14] **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.

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

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

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

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

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

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

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

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

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

[24] —. 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.

# ZałącznikI

## 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 :

* 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