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

życiorysy

**Spis Treści**

[1. Wstęp 14](#_Toc302315481)

[2. Cel i zakres pracy 15](#_Toc302315482)

[3. Wstęp teoretyczny 17](#_Toc302315483)

[3.1. Opis technologii 17](#_Toc302315484)

[3.1.1. Near Field Communication 17](#_Toc302315485)

[3.1.2. Android 19](#_Toc302315486)

[3.1.3. Google Samsung NEXUS S 20](#_Toc302315487)

[3.1.4. Znacznik NFC Forum Type 2 - Mifare Ultralight 20](#_Toc302315488)

[3.1.5. Hibernate 3 22](#_Toc302315489)

[3.1.6. Spring Framework 3 23](#_Toc302315490)

[3.1.7. Oracle 10g Express Edition 24](#_Toc302315491)

[3.1.8. Java 2 Enterprise Edition 24](#_Toc302315492)

[3.1.9. Apache Tomcat 25](#_Toc302315493)

[3.1.10. Architektura REST 25](#_Toc302315494)

[3.1.11. JSON - JavaScript Object Notation 26](#_Toc302315495)

[3.1.12. System kontroli wersji Subversion 1.6 26](#_Toc302315496)

[3.1.13. Eclipse SDK 3.6.2 + Java EE 27](#_Toc302315497)

[3.1.14. Eclipse SDK 3.6.2 + Android SDK 27](#_Toc302315498)

[4. Koncepcja Pracy 28](#_Toc302315499)

[4.1. Wstęp 28](#_Toc302315500)

[4.2. Założenia i budowa systemu 29](#_Toc302315501)

[5. Opis systemu 31](#_Toc302315502)

[5.1. Scenariusz użycia 31](#_Toc302315503)

[5.2. Baza danych 33](#_Toc302315504)

[5.2.1. Opis tabel 35](#_Toc302315505)

[5.2.2. Integralność danych 36](#_Toc302315506)

[5.2.3. Wyzwalacze i sekwencje 37](#_Toc302315507)

[5.3. Opis usługi sieciowej – Web Service 38](#_Toc302315508)

[5.3.1. Cel powstania 38](#_Toc302315509)

[5.3.2. Szczegóły implementacyjne 38](#_Toc302315510)

[5.3.3. Mechanizm Uwierzytelnienia. 39](#_Toc302315511)

[5.3.4. Mechanizm sprawdzania statusu zdarzenia 40](#_Toc302315512)

[5.3.5. Rejestracja nowego zdarzenia 41](#_Toc302315513)

[5.3.6. Wprowadzanie nowej lokalizacji i znacznika 43](#_Toc302315514)

[5.4. Opis aplikacji klienckiej na telefon komórkowy 45](#_Toc302315515)

[5.4.1. Cel powstania 45](#_Toc302315516)

[5.4.2. Założenia 45](#_Toc302315517)

[5.4.3. Komunikacja pomiędzy usługą internetową a użytkownikiem aplikacji 47](#_Toc302315518)

[5.4.4. Tryby uruchomienia 47](#_Toc302315519)

[5.4.5. Odczyt znacznika 48](#_Toc302315520)

[5.4.6. Dobór aplikacji do obsługi znacznika 49](#_Toc302315521)

[5.4.7. Wprowadzenie danych i komunikacja z usługą internetową 49](#_Toc302315522)

[5.4.8. Działanie aplikacji 50](#_Toc302315523)

[5.4.9. Wpływ na zużycie baterii 52](#_Toc302315524)

[5.5. Opis aplikacji administracyjnej na telefon komórkowy 53](#_Toc302315525)

[5.5.1. Cel powstania 53](#_Toc302315526)

[5.5.2. Główne założenia 53](#_Toc302315527)

[5.5.3. Działanie aplikacji 53](#_Toc302315528)

[5.6. Opis panelu administracyjnego 55](#_Toc302315529)

[5.6.1. Cel powstania 55](#_Toc302315530)

[5.6.2. Szczegóły implementacyjne 55](#_Toc302315531)

[5.6.3. Warstwa biznesowa 56](#_Toc302315532)

[5.6.4. Warstwa modelu danych 56](#_Toc302315533)

[5.6.5. Warstwa prezentacji 57](#_Toc302315534)

[5.6.6. Budowa Panelu Administracyjnego 57](#_Toc302315535)

[5.6.7. Role użytkowników oraz zabezpieczenie portalu 59](#_Toc302315536)

[5.6.8. Opis zakładek 59](#_Toc302315537)

[6. Testy 61](#_Toc302315538)

[7. Korzyści płynące z naszego rozwiązania 62](#_Toc302315539)

[8. Podsumowanie 63](#_Toc302315540)

[8.1. perspektywy rozwoju 63](#_Toc302315541)

[8.2. Wnioski 64](#_Toc302315542)

[9. bibliografia 66](#_Toc302315543)

[10. Bibliografia 66](#_Toc302315544)

[11. Załączniki 67](#_Toc302315545)

[SPIS TREŚCI : 70](#_Toc302315546)

Wykaz używanych skrótów

# Wstęp

# Cel i zakres pracy

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

Dzięki infrastrukturze telefonii 3G, oraz złożonemu systemowi monitorowania uzyskujemy pełną i pewną kontrolę nad czasem poświęconym przez pracownika na realizację powierzonych mu zadań.

Kompletna realizacja 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.

W dalszej części pracy przedstawione zostaną nasze rozwiązania na temat doboru poszczególnych technologii, powody zastosowania konkretnych technik implementacji oraz scenariusza użycia kompletnego systemu.

# Wstęp teoretyczny

Rozpoczynając projekt spędziliśmy kilka miesięcy na porównywaniu przeróżnych technologii i doborze optymalnych do realizacji naszego systemu. Pierwsza faza obfitowała w drobne prototypy dzięki którym wyeliminowaliśmy błędne założenia i elementy nadmiarowe, które nie dawały oczekiwanych rezultatów, przy jednoczesnym pogarszaniu jakości efektu końcowego.

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 poszukiwania technologii odpowiednich dla naszego rozwiązania. Komunikacja zbliżeniowa zapewnić nam miała dużą łatwość obsługi systemu przez użytkowników końcowych, dzięki czemu mogliby oni się 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 i relatywnie duże, nieporęczne. **(DOWÓD)**. 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 w jakiś sposób 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 są zawsze wyposażeni 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 akcji.

Łącząc te dwa fakty, czyli komunikację w oparciu o standard RFiD i telefonię komórkową, odkryliśmy istnienie organizacji NFC Forum (DOWÓD), a także, prototypowych modeli telefonów wyposażonych w moduły komunikacji Near Field Communication.

Near Field Communication stanowi grupę technologii krótkodystansowej 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 standardu leżą właśnie w tej technologii.

Komunikacja w NFC oparta jest o zjawisko indukcji magnetycznej powstałej pomiędzy dwiema antenami.

obrazek

### Android

Jednym z kluczowych czynników 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 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 i uniwersalności (DOWODY), 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 i ogromne wsparcie ze strony użytkowników oraz samych twórców systemu, co zaowocowało dostępnością wielu bibliotek rozszerzających możliwości programistyczne.

Jako otwarta platforma Android daje programiście dostęp do obsługi praktycznie wszystkich podzespołów urządzenia i magazynów danych, z drugiej strony ścisłe wersjonowanie 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 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).(DOWÓD)

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

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 **nfc.tech**, w oparciu o którą system dokonuje wstępnego doboru puli aplikacji zdolnych do obsłużenia danego znacznika. Dzięki temu nasza aplikacja reaguje tylko na zbliżenie do modelu znacznika Mifare Ultralight, a gdy jest aplikacją obecnie uruchomioną (pierwszego planu), przejmuje jego obsługę automatycznie, bez ingerencji ze strony użytkownika.

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 w około **10%** przypadków nie 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, jego poziom utrzymuje się poniżej jednego procenta całej konsumpcji.

### Google Samsung NEXUS S

Ponieważ pracownicy mobilni często wyposażeni są w 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 miała aplikacja był już w istotnym zakresie zdeterminowany. Krótkie rozeznanie rynku zakończyło się stwierdzeniem faktu – jedynym ówcześnie dostępnym telefonem spełniającym te wymagania jest Google Samsung Nexus S. Po oficjalnej aktualizacji oprogramowania do wersji Android 2.3.4, telefon spełnił wszystkie złożone w nim oczekiwania.

### Znacznik NFC Forum Type 2 - Mifare Ultralight

Jedną z kluczowych zalet prezentowanego systemu jest jego cena wdrożeniowa i eksploatacyjna. Między innymi z tego powodu zdecydowaliśmy się na wykorzystanie niezwykle tanich 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, przez co wszystko co jest zapisane na znaczniku, może być odczytane przez urządzenie zgodne ze standardem ISO/IEC 14443 **(INNOVA + MIFARE DATASHEET).**

Cechą która stała się krytyczna dla naszego systemu okazał się fakt iż każdy znacznik posiada zapisany na 7 bajtach unikatowy numer seryjny, zwany dalej 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. **(Dodatku A?)**

### 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 tłumaczy 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, wprowadza mechanizm dialektów, który je usuwa oraz ukrywa przed programistą za interfejsem Hibernate.

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 pełni on rolę 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 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 między innymi z pakietu Spring MVC oraz Spring Security, które wykorzystujemy w naszym systemie.

Spring MVC (ang. Model – View – Controller ) jest to implementacja wzorca projektowego Model-Widok-Kontroler. Zakłada on, rozdzielenie warstwy modelu danych oraz logiki biznesowej od warstwy widoku, najczęściej stron JSP pokazujących dane. Przepływem pomiędzy tymi warstwami zarządza kontroler, który także otrzymuje dane wejściowe od użytkowników i zarządza generowaniem odpowiednich widoków.

Spring Security to kolejny pakiet projektu Spring używany w naszym systemie. Jego zadaniem jest zapewnienie bezpieczeństwa panelu administracyjnego. Dzięki udogodnieniom pakietu 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 naszej pracy zdecydowaliśmy się na wykorzystanie systemu zarządzania bazą danych Oracle w wersji 10g Express Edition, stworzonego przez firmę Oracle Corporation. Edycja XE, jest to darmowa wersja serwera bazodanowego Oracle 10g lecz ograniczona funkcjonalnie. Nie jest ona w stanie obsłużyć więcej niż jednego procesora oraz więcej niż 4GB pamięci RAM.

Pomimo swoich ograniczeń, serwer zachowuje swoją 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 ponad 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 to obiektowy język programowania udostępniony 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. W jej skład wchodzą serwety, komponenty EJB (ang. Enterprise JavaBeans), usługi Web Service, pliki JSP, XML oraz inne elementy ściśle związane z wytwarzaniem aplikacji WWW takie jak HTML, 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 implementacje 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).

### Architektura REST

REST ( ang. Representational State Transfer ) jest to wzorzec architektoniczny oparty 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 funkcjonalności jako zasobów łatwo dostępnych poprzez identyfikatory URI (ang Uniform Resources Identifiers), będących najczęściej hiperłączami w sieci. Dzięki mechanizmom zawartym w implementacji możliwe jest przesyłanie zasobów posiadających wiele różnych reprezentacji np. text, xml, json.

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

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

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.

### JSON - JavaScript Object Notation

JSON, jest to prosty format danych, możliwy do odczytania przez człowieka, służący do wymiany informacji. 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.

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

Klasa 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 przez dwie osoby w tym samym momencie, jednocześnie zachowując porządek i możliwość wycofania modyfikacji w sytuacji gdy nie przyniosły one oczekiwanych efektów.

Rozwiązanie to, oparte o architekturę klient-serwer **(Subversion)**, do bezproblemowej i szybkiej pracy wymaga jednak 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 coraz większego zapotrzebowania programistów na narzędzia wspomagające ich prace przy tworzeniu coraz to bardziej zaawansowanego oprogramowania. Z biegiem czasu, Eclipse stał się najpopularniejszym narzędziem pracy developerów na całym świecie.

Projekt Eclipse został stworzony prze firmę IBM w 2001 roku, a w 2004 został udostępniony na zasadzie wolnego oprogramowania. Obecnie, jest to platforma wpierająca najpopularniejsze języki programowania. Dzięki możliwościom rozszerzenia jego funkcjonalności, za pomocą licznych wtyczek (ang. Plug-ins), możliwa jest łatwa współpraca z wieloma frameworkami.

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 przetestowania 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 wymaga korzystania z automatycznie generowanej klasy R (Resource), korzystanie z samego Source Development Kit dostarczonego przez Google jest bardzo uciążliwe. W celu usprawnienia pracy nad aplikacjami mobilnymi zdecydowałem się na skorzystanie ze zintegrowanego środowiska programistycznego (IDE – Integrated Development Environment – **DOWÓD)** Eclipse SDK 3.6.2 połączonego ze wspomnianym zestawem narzędzi Android SDK.

Dzięki temu uzyskałem dostęp do złożonej aplikacji typu debugger, możliwości wglądu w wydruki konsoli systemu Android, 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. Jego zadaniem jest ułatwienie nadzoru nad pracownikami mobilnymi, jednocześnie umożliwiając rzetelne rozliczenie ich czasu pracy. W naszym rozwiązaniu dokonaliśmy odwrócenia klasycznego scenariusza, w którym czytnik RFiD ulokowany jest na stałe w ścianie lub bramce, a użytkownik posiada kartę RFiD. Przedstawiona sytuacja, ma ekonomiczne uzasadnienie w sytuacji, w której mówimy o stałej lokalizacji pracowników i ich dużej liczbie. Problemy pojawiają się, gdy lokalizacji jest wiele (tyle ilu klientów) i mogą się zmieniać, pracownicy stale poruszają się pomiędzy nimi, a pracodawcy zależy na informacjach 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 zarządzającą dostępem.

W sytuacji, w której czytnikiem jest telefon, a zatem urządzenie w które z powodów cywilizacyjnych każdy pracownik tego typu jest wyposażony, nie ma problemu związanego z dodatkowym kosztem (gdyż zawiera się w cenie urządzenia domyślnie używanego przez pracownika), a niezwykle (CENA!) tanie Tagi RFiD, zgodne ze standardem NFC, rozmieszczane zostają na każdym z serwisowanych urządzeń, powodujemy zmianę rozkładu kosztów i zwiększenie szczegółowości informacji (gdyż wiemy dokładnie nad którym zleceniem pracuje w danym momencie dany pracownik).

Dane transmitowane są w czasie rzeczywistym, dzięki czemu koordynator 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 do 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 zapytań POST w formacie JSON. Dane zgromadzone przez usługę, lokowane są w bazie danych, a aplikacja portalu administracyjnego dostarcza możliwość wglądu oraz ich edycji.

Poniżej przedstawiamy schemat systemu:

// to do obrazek

Poniżej zostały przedstawione założenia mające istotny wpływ na poprawne funkcjonowanie 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 której telefonu 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 ludzkimi.
* W przypadku zgłoszenia zawierającego niezarejestrowany znacznik, system zgłosi błąd - z powodu jednoznacznego powiązania zdarzenia z lokalizacją, niemożliwa jest obsługa zgłoszenia zawierającego identyfikator znacznika, nie wprowadzonego do systemu.
* Istnieją 3 grupy dostępu do systemu - system posiada rozbudowany moduł uwierzytelniający użytkowników. Stworzone zostały trzy kategorie dostępu. Każdy użytkownik posiadający login i hasło również posiada 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.

# Opis systemu

## Scenariusz użycia

Aplikacja, może znaleźć zastosowanie w firmach, zajmujących się serwisowaniem urządzeń położonych w różnych zewnętrznych lokalizacjach, należących do zewnętrznych kontrahentów. Przypuśćmy taką sytuację- nasza firma zajmuję się serwisowaniem urządzeń wielofunkcyjnych – kopiarek, skanerów, drukarek.

Pracodawca, chcąc wdrożyć system, instaluje serwer usługi internetowej 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.

Wybrana firma , podpisuje z nami 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, na obudowie którego, umieszczana jest naklejka ze znacznikiem RFiD (NFC) i ewentualnym logiem naszej 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), 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 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.

Istotnym jest zaznaczenie tego, że w jednym momencie dany pracownik może realizować tylko jedno zgłoszenie. Informacja ta, jest przekazywana w postaci czytelnego komunikatu, skierowanego bezpośrednio do pracownika, podczas próby rozpoczęcia kolejnego zlecenia.

Pracodawca lub operator mogą sprawdzić w panelu administracyjnym stany realizacji wszystkich zleceń oraz czas jaki został poświęcony na realizację konkretnej czynności.

## Baza danych

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.

Została ona 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.

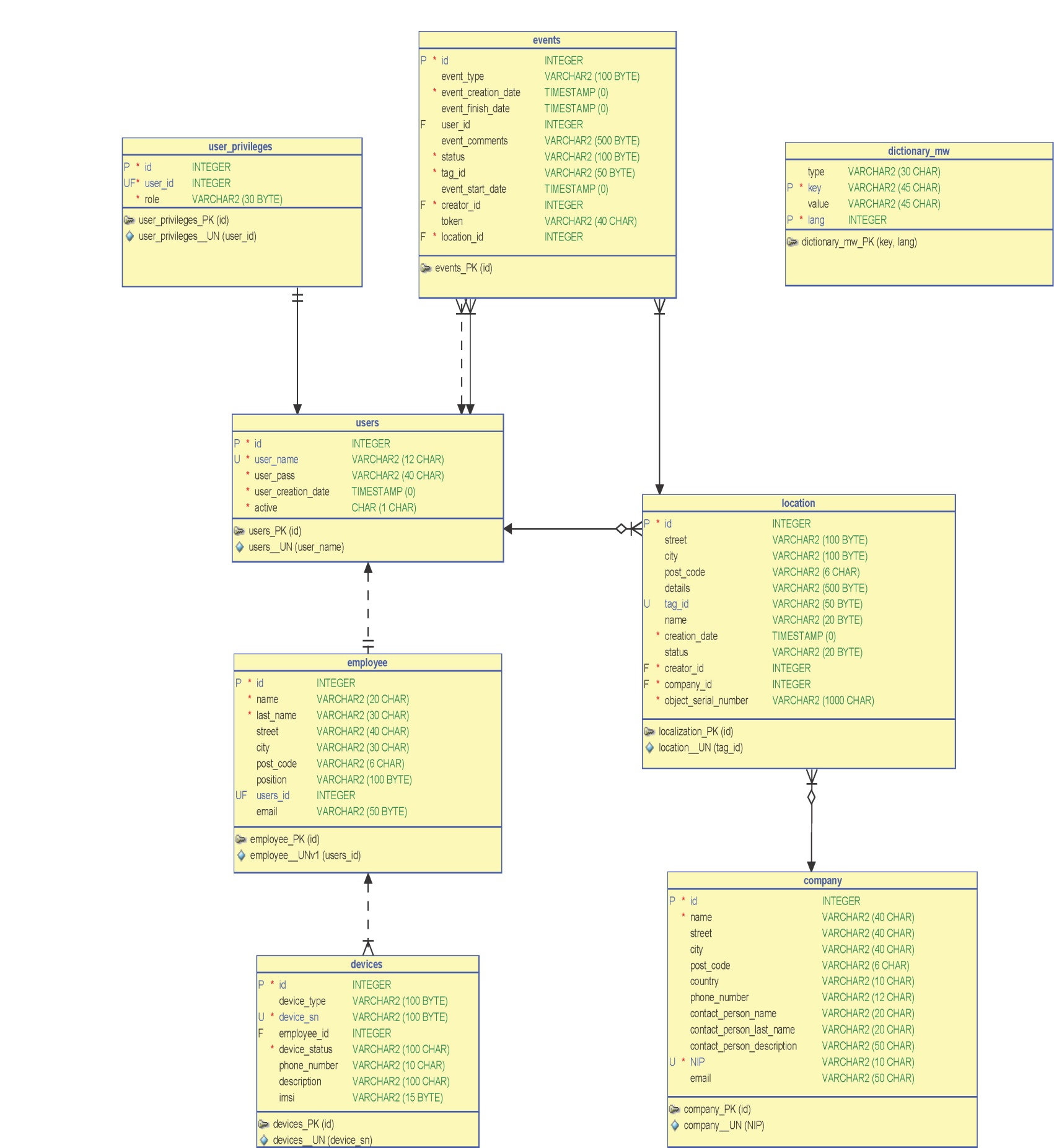
Model relacyjny zakłada grupowanie danych w relacje, które są reprezentowane za pomocą tabel. Są zbiorem rekordów o ustalonej i identycznej strukturze wewnętrznej. Każda z nich posiada nazwę, nagłówek oraz zbiór atrybutów, zawartością relacji z kolei są krotki. Krotka jest jednoznacznie identyfikowana przez klucz główny 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 on 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 Tagu dla 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 Tagu, 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 started lub 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*

*Kod unique index*

### 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

### 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 funkcjonalności pozwalające kolejno na :

* uwierzytelnienie pracownika i umożliwienie mu dalszej pracy z aplikacją mobilną
* sprawdzenie przez aplikacje 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 Tagu

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

* 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 z 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.

Moduł web service 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.

Poniżej opiszę każdą z wyżej wymienionych funkcjonalności a także zaprezentuje przypadki wykorzystania ich w aplikacji.

// nie wiem czy wklejać kod przykładowego odwzorowania hbm.xml i hbm.cfg.xml

### 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 na 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, ze względu na chęć zabezpieczenia się przed możliwością zamiany aparatów, czy też karty SIM pomiędzy pracownikami mobilnymi, chcemy mieć pewność, że dane zdarzenie jest obsługiwane konkretnie przez pracownika będącego właścicielem telefonu.

### 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.

Powodem takiego działania są względy technologiczne implementacji aplikacji na telefon komórkowy. Niestety poprzez wyczyszczenie podręcznej pamięci użytkownika aparatu telefonicznego możliwe jest usunięcie zgromadzonych tam danych odnoszących się do wcześniej zarejestrowanego i rozpoczętego zdarzenia.

Dzięki wyżej wymienionemu mechanizmowi, telefon automatycznie uzupełnia potrzebne mu do dalszego funkcjonowania informacje.

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 użytkownia zajmującego się zdarzeniem, datę utworzenia oraz identyfikator Tagu.

Uproszczony schemat działania rejestracji nowego zdarzenia został przedstawiony na rysunku 1.1



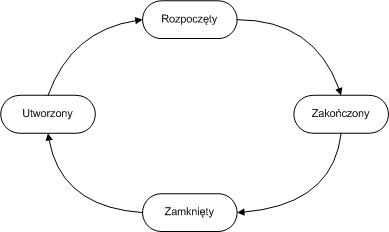
*Rys … Algorytm obsługujący zgłoszenie rejestracji zdarzenia*

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 Tagu jest zarejestrowany w systemie. Jeżeli weryfikacja jest negatywna, aplikacja mobilna otrzymuje odpowiedni status błędu mówiący o braku Tagu w bazie. W przeciwnym razie aplikacja sprawdzi najpierw czy istnieją zdarzenia o statusie „utworzone” dla danego Tagu, 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 przez panel administracyjny oraz token.

Na poniższym schemacie zostało przedstawiony cykl życia nowo zarejestrowanego zdarzenia.



*Rys … Cykl życia zdarzenia*

Poprawić obrazek

### Wprowadzanie nowej lokalizacji i znacznika

Funkcjonalność rejestracji nowej lokalizacji jest przeznaczone 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 lokalizacje 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.

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

* istnieje już w bazie – wtedy aplikacja odmawia jej 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 Tag 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 (DOWÓD), cała logika systemu zrealizowana jest w aplikacji serwerowej, telefon z kolei formułuje zapytania i przetwarza odpowiedzi. Komunikacja w architekturze REST (Representational State Transfer – literatura) 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 sposób niejawny 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:

* 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.) GODZINA ROZPOCZĘCIA

### 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.

W pierwszym wypadku do obsługi znacznika stosowany jest tzw. foreground dispatching, **opisany dokładniej w podrozdziale Dobór aplikacji do obsługi znacznika**. 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.

Dodatkowo aplikacja w pierwszej fazie uruchomienia sprawdza czy posiada zapisane w banku UserPreferences dane logowania i jeśli nie może ich znaleźć, oferuje użytkownikowi wprowadzenie ich bądź opuszczenie aplikacji. Ponieważ przy każdym odczytaniu znacznika dane zdarzenia transmitowane do serwera opatrzone są kompletem danych identyfikacyjnych (z racji konstrukcji 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 też dostępna z poziomu menu aplikacji w każdej chwili. Hasło do logowania przechowywane jest w aplikacji w formie skrótu *SHA-1*, ponieważ…

W tym wariancie aplikacja dokonuje również aktualizacji 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, ten ostatni jest zawsze poinformowany o swoim stanie zarejestrowanym w systemie.

Sytuacją odmienną jest zbliżenie telefonu do znacznika w stanie spoczynku (przy wyłączonej aplikacji). System operacyjny wyświetla wtedy monit z zapytaniem o aplikację która powinna zostać oddelegowana do obsłużenia znacznika, jest to tak zwany system Entent dispatching, **który również zostanie przybliżony w podrozdziale Dobór aplikacji do obsługi znacznika**.

W sytuacji takiej aplikacja uruchamia się tylko w celu odnotowania zdarzenia i po zakończeniu tego procesu i poinformowaniu użytkownika 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.

### Odczyt znacznika

Co prawda 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 i odczyt 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. **(NFC Forum Tag Type 2)**

System operacyjny w momencie wykrycia znacznika w obrębie pola elektromagnetycznego wytwarzanego przez telefon pobiera z niego unikatowy identyfikator i wiadomości NDEF i umieszcza w intencji, która w dalszej kolejności przekazywana jest do odpowiedniej aplikacji w celu ich przetworzenia. 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. **(ANDROID API)**

### 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. **(KSIĄŻKA INTERNETOWA ANDROID)**

W tym celu stworzono dwa systemy wywoływania aplikacji.

Pierwszym z nich jest intent dipatch 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. 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.

Druga z metod obsługi odczytanego znacznika, tzw. foreground dispatching, 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. 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 oraz IMSI (rozwinięcie skrótów), 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 | (IMSI Guidelines v12!!) 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 (hmmm???), 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 „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 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 przez operatora.

Obrazki

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.

### Wpływ na zużycie baterii

Podczas testów wydajnościowych 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).

## Opis aplikacji administracyjnej na telefon komórkowy

### Cel powstania

Aplikacja administracyjna używana jest do rejestracji lokalizacji usługowych przez 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 poprawnej autoryzacji w systemie 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 orientacyjną nazwę kontrahenta i jego numer NIP. Pola formularza podlegają walidacji, a orientacyjna 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 i 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 użytkownika 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” rozpoczyna walidację formularza.

Po pomyślnym przejściu walidacji pól, aplikacja zwraca się do użytkownika z prośbą o zbliżenie znacznika i korzystając z systemu foreground dispatching (ANDROID) uzyskuje najwyższy priorytet przy jego obsłudze **(\*niezrozumiałe**). W następnym kroku, jeśli użytkownik zbliży znacznik, 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. Z tego powodu w procesie rejestracji, na znaczniku zapisywana jest tylko nazwa aplikacji, a cała rejestracja zachodzi po stronie usługi internetowej.

Obrazki:

W chwili obecnej, z przyczyn finansowych (nie mamy dostępu do nieograniczonej liczby znaczników), funkcja blokowania zapisu na znaczniku jest wyłączona. Za pośrednictwem interfejsu użytkownik otrzymuje stosowny komunikat i może kontynuować rejestrację pozostałych urządzeń. W przeciwnym wypadku system powiadamia o niepowodzeniu i jego przyczynie i 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.

Użytkownikiem panelu administracyjnego są osoby zarządzające czasem pracy pracowników mobilnych. Dzięki dostarczonym funkcjonalnością, mogą 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 Enetrpriese 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. Ich lista została przedstawiona poniżej.

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

Moduł 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 odpowienich 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.service jest modułem, który znajduję się na granicy pomiedzy 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, lokalizacje czy też firmę. 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 edycje danych bez odpowiednich uprawnień.

### 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 funkcję 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 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, poprzez ustawienie go w statusie zamknięte, dodając opcjonalny dodatkowy komentarz.

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 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 uprawnieniem 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, z tego poziomu, dodanie nowego zdarzenia do systemu. W takim przypadku, pracownik zostaje przeniesiony do formularza, przez który wprowadzi typ zdarzenia, komentarz oraz wybierze lokalizacje usługi.

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

// screen

Zakładka „Lokalizacje” ma za zadanie wyświetlenie użytkownikowi wszystkich lokalizacji wprowadzonych do systemu. Pozwala na wgląd do szczegółowych danych o wybranym miejscu. Operator ma możliwość ich edycji jak i dodawania nowej pozycji do systemu.

// screen

Po wybraniu pozycji „Urządzenia” z menu, zostaje przedstawiona lista urządzeń, 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”. Po jej wybraniu wyświetlana jest lista 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 wyboru, 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”.

// screen

# Testy

York

Lodówka

Miasto i 3G

Wifi

Różne scenariusze

# 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. 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

# bibliografia

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

[2] Minter Dave, Linwood Jeff, *Hibernate Od Nowicjusza do Profesjonalisty*, Power Net, 2007.

[3] Price Jason, *Oracle Database 11g I SQL Programowanie,* Helion, Gliwice 2009.

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

# Bibliografia

1. **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.

# Załączniki

**WYMAGANIA REDAKCYJNE DOTYCZĄCE PRAC DYPLOMOWYCH INŻYNIERSKICH I MAGISTERSKICH**

**I. Wymagania edytorskie**

1. Praca dyplomowa powinna być wydrukowana na papierze białym formatu A4, tekst powinien być wydrukowany w kolorze czarnym, zalecany jest druk dwustronny, z wyjątkiem strony tytułowej, streszczeń i życiorysu (-ów).

2. Marginesy wszystkich stron z wyjątkiem strony tytułowej: lewy 3 cm, prawy 2 cm, dolny 2 cm, górny 2 cm, strona tytułowa: margines lewy 3 cm, prawy 1,5 cm, górny 1,5 cm i dolny 1,5 cm.

3. Czcionka tekstu podstawowego Times New Roman, prosta, 12 ppt,

4. Interlinia 1,5,

5. Tekst wyrównany (wyjustowany) do prawej,

6. Wcięcie na początku nowego akapitu: 1 cm, nie należy stosować wcięcia w pierwszym akapicie rozdziału , podrozdziału itd.

7. Pomiędzy sąsiednimi akapitami nie zwiększać odstępu, tj. zachować interlinię 1,5.

8. Nowy rozdział powinien zaczynać się od nowej strony,

9. Tytuł podrozdziału, punktu, podpunktu itd, nie może kończyć strony,

10. Rozdział, podrozdział itd. nie może zaczynać się od rysunku, tabeli ani wykresu,

11. Między wierszem poprzedzającym i następującym po tytule rozdziału, podrozdziału itd., a tytułem interlinia powinna wynosić 2,

12. Tytuły:

- rozdziałów powinny być pisane czcionką pogrubiona prostą Times New Roman 16 ppt, dużymi literami z numeracją arabską rozdziału np.:

**4. ANALIZA SYGNAŁU**

- podrozdziałów czcionką pogrubioną prostą Times New Roman 12 ppt, dużymi literami z numeracją arabską, np.

**4.1. ANALIZA SYGNAŁU**

- tytułu punktów czcionką pogrubioną prostą Times New Roman 12 ppt, małymi literami z numeracją arabską, np.

**4.1.1. Analiza sygnału**

- tytułu podpunktu czcionką normalną prostą Times New Roman 12 ppt, małymi literami z numeracją arabską, np.

4.1.1.1. Analiza sygnału

- w przypadku stosowania piątego poziomu numeracji należy stosować czcionkę kursywę Times New Roman 12 ppt, małe litery z numeracją arabską, np.

4.1.1.1.1. *Analiza sygnału*

13. Po tytułach rozdziałów, podrozdziałów, podpunktów, rysunków, tabel i wykresów nie stawia się kropek.

14. Strona tytułowa powinna być zgodna z podanym wzorem. Informacja o opiekunie naukowym jest opcjonalna i występuje tylko wtedy, gdy kierujący pracą nie ma stopnia naukowego doktora.

15. Numeracja stron powinna być ciągła, poczynając od strony tytułowej, ale pierwszą stroną z umieszczonym numerem powinna być druga strona spisu treści. Numery stron powinny znajdować się w prawym dolnym roku. Dopuszcza się używanie nagłówków i stopek. Gdy używana jest stopka to numer strony powinien być umieszczony wewnątrz niej. Nagłówków nie wstawia się na stronie tytułowej, stronach ze streszczeniami, stronie z życiorysem oraz na pierwszej stronie wstępu, pierwszej stronie nowego rozdziału, pierwszej stronie wykazu literatury oraz pierwszych stronach załączników.

16. Odnośniki powinny być pisane czcionką prostą Times New Roman 10 ppt, podobnie podpisy pod rysunkami, tabelami i wykresami, stosować czcionkę normalnej grubości.

17. Inne kolory druku niż czarny mogą być stosowane tylko tam, gdzie jest to uzasadnione czytelnością tekstu, wykresu bądź rysunku,

18. W podpisach stosować pojedynczą interlinię. Podpis rysunku powinien znajdować się pod rysunkiem, centralnie, albo z wyjustowaniem do lewej strony. Tak samo w przypadku wykresów. Podpis tabeli może być umieszczany zarówno nad tabelą, jak i poniżej niej, ale jednakowo w całej pracy. Numeracja tabel i rysunków jest rozłączna. Wykresy mogą być uznawane za rysunki, jeżeli jest inaczej to ich numeracja powinna być osobna. Proponuje się stosować numerację dwupoziomową, tzn. od początku w każdym rozdziale (np. Rys. 2.1, Rys. 4.1, Tabela 3.1). Podpis nie może być przeniesiony na następną stronę.

20. Odnośniki do literatury umieszczać w nawiasach kwadratowych, każda pozycja literaturowa umieszczana w wykazie literatury musi być tak opisana, aby dało się ją bez trudności odnaleźć, w szczególności musi być podane nazwisko autora (w przypadku materiałów firmowych nazwa firmy), tytuł, wydawnictwo, rok wydania, numer czasopisma itd., w przypadku materiałów z Internetu należy podać miesiąc i rok zaglądania na stronę,

21. Wszystkie używane w pracy skróty powinny wraz z ich rozwinięciami znajdować się w wykazie skrótów, skrót użyty w zdaniu musi być poprzedzony przynajmniej jednym słowem z nim związanym (np. *W systemach SDH używa się* …, a nie *W SDH używa się* …).

**II. Układ pracy**

Kolejność stron w pracy dyplomowej powinna być następująca:

1. Strona tytułowa

2. Streszczenie po polsku (maksymalnie 1 strona)

3. Streszczenie po angielsku (maksymalnie 1 strona)

4. Życiorys (-y)

5. Spis treści

6. Wykaz używanych skrótów

7. Wstęp

8. Rozdział 1, 2 itd.

9. Podsumowanie i wnioski końcowe

10. Literatura

11. Załączniki

**III. Liczba egzemplarzy pracy**

Wymagane jest złożenie jednego egzemplarza pracy dyplomowej w Sekretariacie Dydaktycznym IT. Jeżeli wymaga tego kierownik pracy (opiekun) to u niego składa się dodatkowy egzemplarz pracy.

**IV. Obowiązkowe załączniki do pracy dyplomowej**

Pyta CD

## SPIS TREŚCI :

1. Wstęp
2. Cel i zakres pracy
3. wstęp teoretyczny
   1. opis technologii
      1. ~~android~~
      2. ~~nexus~~
      3. ~~oracle~~
      4. nfc
      5. ~~Mifate Ultralight~~
      6. ~~J2EE~~
      7. ~~Spring~~
      8. ~~Hibernate~~
      9. ~~Jersey~~
      10. ~~Restful application – json~~
      11. eclipse
   2. przyczyny doboru ?
4. ~~koncepcja pracy~~
5. opis systemu
   1. ~~scenariusz użycia~~
   2. opis telefonu
      1. ~~aplikacja rejestrująca~~
      2. aplikacja administratora
   3. ~~opis Web-service – usługa internetowa~~
      1. ~~4 pod serwisy~~
   4. ~~opis bazy danych~~
   5. opis panelu administracyjnego
6. korzyści płynące z rozwiązania
7. podsumowanie
8. wnioski

bibliografia