# WSTĘP

Centrum powiadomień ratowniczych to system, dzięki któremu każdy z nas może zaalarmować służby ratownicze w razie nagłej potrzeby. Polega on na pracy operatorów, którzy odbierają takie połączenie, uzyskują informację od rozmówcy, a następnie powiadamiają odpowiednie organy, które jak najszybciej starają się zainterweniować. Cała procedura trwa do kilku minut.

W sytuacjach zagrożenia życia często ważna jest każda sekunda. Przybycie służb ratunkowych chwilę później może skończyć się tragicznie. Rozwój technologiczny wpływa na prawie każdy aspekt naszego życia. Nie inaczej jest z procedurą obsługi zgłoszeń alarmowych. Każdą rzecz można ulepszyć, poprawić oraz przyspieszyć, co w tym konkretnym temacie ma kluczowe znaczenie.

Aktualnie operatorzy otrzymują pewne informacje automatycznie. Jeśli połączenie jest z telefonu stacjonarnego są to: imię, nazwisko oraz adres osoby, na którą dany telefon jest zarejestrowany. Jeśli jest to telefon komórkowy z aktualnie włączoną lokalizacją GPS to również zdarza się, że współrzędne geograficzne.

Można jednak pójść kilka kroków dalej i zawsze przekazywać współrzędne. Zautomatyzować dostarczenie do operatora pewnych informacji. Inaczej mówiąc skrócić trwanie procedury do minimum. Pozostawiając oczywiście możliwość korzystania z aktualnych procedur. Ewentualnie można połączyć nowe rozwiązania z klasyczną rozmową telefoniczną. Te trzy możliwości doprowadzą do skrócenia bądź wyeliminowania naszego kontaktu z dyspozytorem.

# 1.1. Cel pracy

Celem niniejszej pracy jest opracowanie systemu powiadomień ratowniczych składającego się z dwóch części:

* aplikacji mobilnej na telefony i tablety z systemem Android służącej do przekazania wszystkich informacji potrzebnych w procedurze zgłoszenia. Powinna być jak najprostsza w użyciu i umożliwiać automatyczne przekazanie wszystkich informacji bądź cząstkowych z ewentualną telefoniczną korektą.
* aplikacji webowej służącej operatorom do otrzymywania powiadomień, kolejkowania ich, akceptowania oraz edycji.

Całość powinna być prosta i jak najszybsza w użyciu tak dla jednej, jak i drugiej strony.

## 1.2. Zakres pracy

TUTAJ będzie lista rozdziałów z krótkim opisem zrobię ją na końcu wtedy kiedy spis treści

## 1.3. Analiza rynku

Aktualnie na rynku brak dostępnych aplikacji tego typu. Istnieją jedynie małe aplikacje umożliwiające kontakt ze służbami takimi jak straż pożarna, pogotowie ratunkowe, policja bądź GOPR. Centrum powiadamiania ratunkowego bazuje praktycznie w całości na rozmowie telefonicznej z dyspozytorem, który decyduje co dalej zrobić.

Problemem takich systemów jest konieczność wdrożenia systemu odbioru zgłoszeń w centrum powiadamiania ratunkowego. Jest to droga i długa procedura wymagająca zaangażowania wszystkich centrów. System działa tak, że w razie awarii jednego centrum lub przekroczenia liczby połączeń, następuje przekierowanie do CPR znajdującego się w innym województwie. Wymagane jest też jak najlepsze odrzucenie fałszywych lub niezasadnych wezwań. Niestety takie połączenia w tym momencie stanowią około 46% wszystkich, co jest ogromną liczbą.

## 1.4. Krótki opis tworzonego systemu

System w razie nagłej potrzeby skontaktowania się z centrum alarmowym, pozwala użytkownikowi zrobić to jak najszybciej, po wcześniejszym pobraniu aplikacji i jej zainstalowaniu na urządzeniu mobilnym. Korzystający musi się zarejestrować i potwierdzić swoją tożsamość, następnie jest poproszony o wyrażenie zgody na korzystanie z modułu GPS. Od teraz w razie potrzeby może włączyć aplikację i poprzez odpowiadanie na najważniejsze pytania, przyciskając odpowiednie przyciski na ekranie urządzenia, system skompletuje informację, a następnie wysyła je do systemu odbioru powiadomień ratowniczych. Gdzie powiadomienie dostarczane są do kolejki i kiedy operator dojdzie do tego konkretnego wezwania, będzie mógł zdecydować czy je odrzucić, oddzwonić w celu uzupełnienia danych lub zaakceptować i powiadomić odpowiednie służby.

Użytkownik w trakcie korzystania może w każdym momencie zakończyć działanie aplikacji i zadzwonić bezpośrednio do operatora. Może wybrać czy chce przesłać wypełnione już informacje razem z połączeniem telefonicznym, czy odrzucić to co wcześniej uzupełnił. Daje to możliwość wykorzystania starych jak i nowych metod, które udostępnia system- jednocześnie.

# Analiza zagadnienia systemu powiadomień ratowniczych

## 2.1 Definicja

System przekazujący zgłoszenia alarmowe w celu zaangażowania odpowiednich służb ratowniczych.

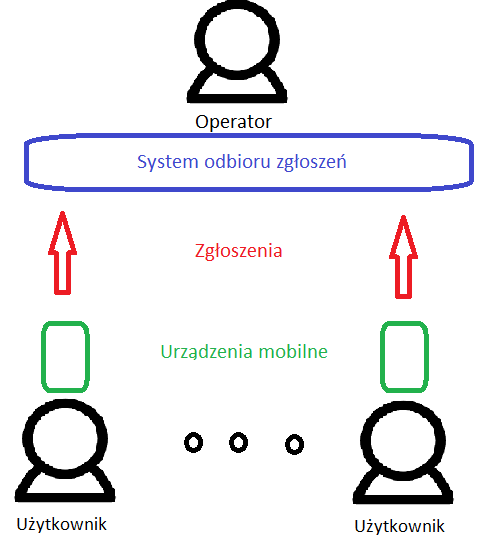
## 2.2 Cechy

Cechy systemu powiadomień ratowniczych:

* Najważniejszą cechą takiego systemu jest szybkość przekazania zgłoszenia do służb ratowniczych. Cała procedura powinna trwać jak najkrócej, aby zwiększyć szanse powodzenia akcji ratowniczej.
* Wszystkie zgłoszenia powinny wylądować w kolejce tak, aby żadne zgłoszenie nie zostało pominięte. Obsługa dokonywana jest w kolejności wpłynięcia zgłoszenia.
* Zautomatyzowanie przekazywania informacji jak dane zgłaszającego,
* Wskazanie dokładnego miejsca zdarzenia poprzez wykorzystanie sygnału GPS do określenia współrzędnych geograficznych, celem uniknięcia podania przez świadka bądź ofiarę błędnych informacji.
* Ograniczenie pracy operatora odbierającego zgłoszenia, tak aby mógł obsłużyć jak największą ilość zgłoszeń.
* Dokonanie ewentualnej korekty już zaakceptowanej procedury przez operatora.
* System winien być możliwie jak najprostszy, zachowując jednocześnie pełną funkcjonalność. Obsługa i dla zgłaszającego jak i operatora powinna polegać na kilku kliknięciach i ewentualnej rozmowy telefonicznej.
* Wymagania systemowe powinny być jak najmniejsze, tak aby jak największa liczba osób mogła skorzystać z rozwiązania.
* Wszystkie zgłoszenia powinny zostać zarchiwizowane. Jest to wymagane prawnie. Przydaje się to w sytuacjach, kiedy wymagane jest potwierdzenie wykonania całej procedury.
* Zbieranie statystyk jest również bardzo ważne. Pozwala to w przyszłości lepiej organizować całą strukturę centrum powiadamiania, jak również przesuwać zasoby z miejsc cechujących się mniejszym zapotrzebowaniem służ ratunkowych, do miejsc wymagających większej uwagi.

## 2.3. Schemat

Poniżej przedstawiony został schemat systemu powiadomień ratowniczych. Wielu użytkowników poprzez urządzenia mobilne wysyłają zgłoszenia do systemu odbioru zgłoszeń. Tam już operator zajmuje się akceptacją oraz uruchomienie procedur powiadamiających odpowiednie służby.



## 2.4. Zastosowanie systemu

System powiadomień ratowniczych ma zastosowanie w gamie różnych nieszczęśliwych sytuacji. Od pożaru, wypadku samochodowego, choroby, a na napadzie kończąc. Operator w miarę potrzeb może powiadomić Państwową Straż Pożarną, Państwowe Ratownictwo Medyczne bądź Policję. System powinien służyć w wyjątkowych sytuacjach, tak aby przyspieszyć procedurę wezwania odpowiednich służb.

## 2.5. Wymagania

Ten rozdział jest podzielony na dwie części. Jedna z nich opisuje wymagania funkcjonalne, a druga niefunkcjonalne.

## 2.5.1. Wymagania

Patrząc na funkcjonalność, tak i systemu jak i aplikacji, najważniejszymi podmiotami są: pracownik – operator, użytkownik urządzenia mobilnego oraz administrator systemu powiadomień.

Poniższa tabela przedstawia wymagania funkcjonalne systemu.

|  |  |  |
| --- | --- | --- |
| Nr | Nazwa | Opis |
| 1 | Tworzenie kont | Tworzenie kont dla nowych operatorów oraz administratorów systemu |
| 2 | Edycja kont | Edycja kont operatorów i administratorów |
| 3 | Usuwanie kont | Usuwanie kont osób, które już nie pracują |
| 4 | Zmiana hasła | Zmiana hasła dla istniejących kont |
| 5 | Przeglądanie kont | Przeglądanie listy kont oraz szczegóły każdego konta |
| 6 | Przypisywanie ról do kont | Przypisywanie ról do kont celem ograniczenia widoczności funkcjonalności |
| 7 | Logowanie do systemu | Logowanie do systemu |
| 8 | Wylogowanie z systemu | Wylogowanie z systemu |
| 9 | Wyświetlanie listy zgłoszeń | Wyświetlenie listy zgłoszeń z najważniejszymi informacjami oraz kilka informacji podsumowujących listę |
| 10 | Odbieranie zgłoszeń | Przekazywanie do obsługi przez siebie kolejnego wolnego zgłoszenia z kolejki przez operatora |
| 11 | Edycja zgłoszeń | Możliwość edycji odebranego zgłoszenia |
| 12 | Wyświetlanie lokalizacji | Wyświetlanie lokalizacji w postaci znacznika na mapie Google |
| 13 | Odrzucanie zgłoszeń | Odrzucanie zgłoszeń, które zostały uznane przez operatora za nie prawidłowe |
| 14 | Przekazywanie służbom | Możliwość przekazania odebranego zgłoszenia do odpowiednich służb ratunkowych |
| 15 | Oddzwanianie do zgłaszającego | Opcja oddzwonienia na nr telefonu, z którego było wysyłane zgłoszenie |
| 16 | Prezentacja metryk systemu | Zaprezentowane metryki systemu takie jak zużycie pamięci, wątków, aktywnych zapytań HTTP, poprawnych jak i błędnych itp. celem analizy poprawności działania systemu i jego wydajności |
| 17 | Historia zgłoszeń | Pełna lista wszystkich zgłoszeń we wszystkich statusach |
| 18 | Statystyki zgłoszeń | Wykresy statystyczne na podstawie szczegółów zgłoszeń |
| 19 | Zmiana języka | Przełączanie języka między polskim, a angielskim |

W tabeli poniżej przedstawione zostały wymagania funkcjonalne aplikacji mobilnej.

|  |  |  |
| --- | --- | --- |
| Nr | Nazwa | Opis |
| 1 | Logowanie przez Facebook | Funkcjonalność umożliwiająca szybką rejestracje do aplikacji mobilnej |
| 2 | Prezentacja danych użytkownika | Wyświetlenie imienia i nazwiska zalogowanej osoby oraz jej zdjęcia profilowego |
| 3 | Wylogowanie | Możliwość wylogowania z aplikacji mobilnej |
| 4 | Dzwonienie na nr 112 | Dostępna opcja wybierania numeru alarmowego 112 |
| 5 | Wielokrokowy formularz zgłoszenia | Funkcjonalność pobierania informacji od użytkownika celem dokonania zgłoszenia ratunkowego |
| 6 | Przerwanie formularza | Możliwość zakończenia wypełniania formularza w każdym możliwym momencie i przesłanie już uzupełnionych informacji do operatorów |
| 7 | Pobranie informacji o lokalizacji | Uzupełnienie zgłoszenia o informacje lokalizacji urządzenia mobilnego |
| 8 | Wysłanie zgłoszenia | Przesłanie zgłoszenia do kolejki RabbitMQ znajdującej się na maszynie serwera |
| 9 | Mechanizm odpowiedzi na zgłoszenie | Otrzymanie odpowiedzi ze statusem wysłanego zgłoszenia |

#### 2.5.2 Wymagania niefunkcjonalne

W ramach wymagań niefunkcjonalnych znajdują się ograniczenia, które system i aplikacja powinny spełniać, aby mogły spełniać odpowiednie funkcje.

Architektura:

* Budowa systemu powiadomień ratowniczych powinna być typowa dla klasy Enterprise. Baza danych jak i wszelkie komponenty powinny znajdować się po stronie serwera serwującego dostęp do określonych funkcjonalności (w tym przypadku przez przeglądarkę)
* Aplikacja powiadomień ratowniczych powinna działać samodzielnie i nie mieć dostępu do danych na serwerze, a jedynie korzystać z udostępnionej przez serwer kolejki
* Architektura powinna zapewniać krótki czas i niskie koszty rozbudowy

Bezpieczeństwo:

* Autoryzacja dostępu do systemu – wymagana rejestracja, przechowywanie hasła w postaci zaszyfrowanej w bazie danych, ustawianie roli przez administratora
* Transakcyjność wszelkich operacji na bazie danych
* Informacje o błędach powinny być wyświetlane
* Dostęp do aplikacji jedynie po zalogowaniu
* Newralgiczne miejsca powinny posiadać napisane jak najdokładniejsze testy, aby uniknąć błędów. Szczególnie podczas rozbudowy

Prostota użycia:

* Korzystanie z systemu i z aplikacji powinno być intuicyjne oraz powinno wymagać jak największej szybkości działania
* Powinna być zapewniona łatwość aktualizacji

Dostępność:

* System powinien działać na wielu przeglądarkach i w wielu wersjach
* Aplikacja mobilna powinna funkcjonować na jak największej liczbie wersji systemu android i na różnych urządzeniach

#### 2.6. Przypadki użycia

Diagram przypadków użycia pokazuje w sposób wizualny funkcje i wymagania, które będą świadczone przez system i aplikacje mobilną określnym aktorom.

Diagram przypadków użycia

# Słownik pojęć

W celu ułatwienia zrozumienia pewnych zagadnień poniżej zaprezentowany jest słownik wybranych pojęć:

TUTAJ będzie słownik pojęć

# Zastosowane technologie i narzędzia

Celem tego rozdziału jest omówienie wszelkich narzędzi i bibliotek wykorzystanych do opracowania systemu oraz opis użytych technologii.

## 4.1 Narzędzia

### 4.1.1 IntelliJ IDEA

Jest to zintegrowane środowisko programistyczne wyprodukowane przez firmę JetBrains. Podstawą jest język JAVA, pomimo, że IDEA współpracuje z ogromną ilością języków, technologii i frameworków. Wielką zaletą jest indeksowanie plików, co pozwala na bardzo szybkie i wygodne przeszukiwanie kodu. Praca w tym środowisku jest bardzo wygodna i przejrzysta. Całość można również poprawić instalując dodatki i wtyczki.

### 4.1.2. Android Studio

Zintegrowane środowisko programistyczne dla platformy Android. Opracowane przez twórców IDEA. Bazuje na wcześniejszym tworze JetBrains, wykorzystując wszystkie najlepsze cechy tego środowisko, orientując je pod wykorzystanie przy tworzeniu aplikacji na urządzenia mobilne. Umożliwia użycie wbudowanego emulatora systemu Android, co bardzo ułatwia proces deweloperski.

### 4.1.3. Repozytorium GIT

GIT to darmowy system kontroli wersji kodów źródłowych. Pozwala zarchiwizować i kontrolować wszelakie zmiany, które zostały dokonane w projekcie, umożliwiając łączenie, usuwanie i przywracanie wcześniejszych wersji oraz tworzenie gałęzi, na których mogą znajdować się różne wersje systemu.

### 4.1.4. Maven

Jest to narzędzie, które automatyzuje budowanie projektów napisanych na platformie JAVA na podstawie pliku POM(Project Object Model), czyli pliku w formacie XML zawierającymi między innymi zależności czy wersje generowanego oprogramowania. Został stworzony przez fundację Apache.

### 4.1.5. Gradle

Jest to zbiór open-source’owych narzędzi do budowania oprogramowania dla kilku języków. Został zbudowany na bazie Apache Ant i Apache Maven i opera się na języku DSL(domain-specific language), czyli języku dziedzinowym, który jest językiem składającym się z paradygmatów programowania mających na celu rozwiązanie pewnego problemu. W tym przypadku budowania aplikacji z kodów źródłowych.

## 4.2 Technologie

### 4.2.1. Java

Głównym językiem wykorzystanym do tworzenia obu części pracy jest język Java. Jest to obiektowy język programowania stworzony przez firmę Sun Microsystems. Jest jednym z najpopularniejszych na rynku. Na sam język składa się kilkanaście kluczowych koncepcji, na które składa się sześć najważniejszych:

* obiektowość – programy definiuje się jako obiekty, których elementami są stan oraz zachowanie. Sam program jest zbiorem takich obiektów, które współpracują ze sobą w celu wykonywania zadań.
* dziedziczenie – polega na przejmowaniu właściwości i zachowań jednego obiektu przez drugi.
* niezależność architektury – kod Javy kompiluje się do kodu pośredniego, który jest niezależny od systemu operacyjnego i komponentów urządzenia.
* sieciowość - cecha umożliwiająca programowanie rozproszone łącząc programy z różnych źródeł, a nawet napisanych w innych językach.
* niezawodność – niezawodność Javy wywodzi się z rozpoznania największych błędów programistycznych języka C++ .
* bezpieczeństwo – między innymi dzięki systemowi wyjątków, który pozwala ograniczyć ilość błędów popełnianych przez programistę.

### 4.2.2. Spring framework oraz model MVC

Szkielet tworzenia aplikacji. Powstał w 2003 roku przez osoby, które założyły później firmę Interface21. Jest to framework, który składa się na wiele zagadnień technicznych. Usprawnia budowanie aplikacji na platformie Java Enterprise Edition. Pozwala tworzyć złożone systemy bez konieczności tworzenia modeli programowania, jak to było we wcześniejszych rozwiązaniach. Zawiera wiele szablonów pomagających w programowaniu aspektowym, przy dostępie do danych bądź transakcyjnością. Posiada kontener, który pozwala na wstrzykiwanie zależności, czyli szybsze i prostsze implementowanie cząstek aplikacji. Spring posiada szablon Model-Widok-Kontroler. W tej pracy jest on podstawą nie tylko w kwestiach związanych z warstwą niewidoczną dla użytkownika, ale również został wykorzystany w części frontendowej.

MVC to wzorzec organizujący aplikacje z graficznym interfejsem. Sam model bazuje na trzech częściach:

* model – czyli reprezentacja logiki aplikacji. Składa się na obiekty reprezentowane w widoku oraz te niewidoczne dla użytkownika.
* widok – opisuje formę wyświetlania części modelu. Tworzy interfejs użytkownika, a w jego skład wchodzą podwidoki.
* kontroler – służy do pobierania danych wejściowych, opracowania ich, reakcji na wszelkie działania użytkownika oraz odświeżania widoków.



### 4.2.3. Bazy danych H2, Oracle oraz Hibernate framework

W pracy do obsługi baz danych wykorzystano framework Hibernate. Współpracuje on z wieloma rodzajami baz SQL. Na potrzeby systemu wykorzystano dwie: H2 oraz Oracle. H2 jest to bardzo szybka, darmowa baza działająca w pamięci i pozwalająca na dostęp poprzez przeglądarkę lub poprzez odpowiednie narzędzie, np. wbudowany moduł dostępu do baz w IntelliJ IDEA. Tę bazę wykorzystano w celach deweloperskich. System korzysta z niej tylko przy uruchomieniu aplikacji z profilem „dev”. Jest to spowodowane tym, że H2 potrzebuje mniej zasobów i uruchamia się oraz działa szybciej. Jest preferowana przy pracy nad aplikacją. Z kolei system podczas pracy produkcyjnej wykorzystuje bazę danych Oracle, która jest stabilniejsza z dużą ilością danych oraz przoduje na rynku w kategorii najmniej awaryjnej i pewnej bazy danych.

#### 4.2.3.1. H2

Darmowa baza danych napisana w języku Java. Może zostać uruchomiona z poziomu aplikacji bądź przy wykorzystaniu serwera. Korzysta z dialektu SQL. Działa na wszystkich platformach, na których została zainstalowana Java. Posiada wbudowaną ochronę przed SQL Injection, czyli wstrzykiwaniem kodu w wywoływaniu zapytań, co mogłoby zostać wykorzystane na wiele sposobów. Ciekawą opcją jest możliwość zarządzania bazą poprzez konsolę używając do tego celu zwykłej przeglądarki internetowej.

#### 4.2.3.2. Oracle Database

Jest to system, który zarządza relacyjnymi bazami danych stworzony przez firmę Oracle. Wykorzystuje język zapytań SQL. Dodatkowo można skorzystać z wbudowanego języka PL/SQL, który służy do tworzenia procedur składowych, które obudowują język SQL, pozwalając tworzyć dużo bardziej skomplikowane skrypty bazodanowe. Oracle jest jedną z najpopularniejszych, jeśli nie najpopularniejszą, bazą danych wykorzystywaną w wielu komercyjnych systemach.

#### 4.2.3.3. Hibernate

Jest to framework, który realizuje dostęp do danych. Najprościej mówiąc wykonuje translację danych z relacyjnej bazy danych na obiekty. Opis struktury, jaką ma przetworzyć, opiera się na języku XML. Dzięki niemu można, bez problemu, wgrywać obiekty z pamięci programu bezpośrednio do bazy danych. Dużym plusem jest również to, że Hibernate zwiększa wydajność operacji poprzez buforowanie i tworzenie jak najmniejszej ilości przesyłanych zapytań. Jest darmowym frameworkiem. Z wszystkich frameworków na rynku to on posiada największe API.

#### 4.2.4. Angular JS

Stworzony przez Google otwarty framework, który opiera się na popularnym JavaScript. Usprawnia tworzenie aplikacji internetowych wykorzystując model MVC. Najważniejszą funkcją Angular jest dwukierunkowe wiązanie co pozwala na wyłapanie przez framework zmian w modelu, od razu modyfikując widok, do którego wykorzystywany jest język HTML. Angular posiada bardzo dużo ciekawych opcji jak watch’e czy dyrektywy. Można stworzyć widok, wykorzystując do tego kilka mniejszych, co ułatwia modyfikacje tylko konkretnego elementu. Do przełączania między stanami wykorzystać można kilka możliwych routingów, na które pozwala Angular.

#### 4.2.5. HTML5

Język wykorzystywany do tworzenia i prezentacji widoków stron www. Posiada on kompatybilność wsteczną, dzięki czemu można korzystać z nowych, jak i starych rozwiązań. HTML polega na znacznikach, których bazę wersja 5 bardzo wzbogaca.

#### 4.2.6. Bootstrap framework i CSS3

Bootstrap to framework CSS zawierający wiele zestawów narzędzi interfejsu graficznego aplikacji internetowych. Bazuje na HTML oraz CSS. Dzięki niemu stylizacja elementów strony jest prostsza i przejrzystsza.

CSS3 to rozwinięcie poprzedniczek. Wprowadza takie cechy jak moduły. Moduł zawiera nowe możliwości i jest po prostu rozszerzeniem wcześniejszych elementów. Zaletami takiego wykorzystania stylizacji strony to rozdzielenie warstw prezentacji od danych. Tak, aby można było modyfikować samą stronę wizualną.

#### 4.2.7. JUnit i Protractor

JUnit to narzędzie wykorzystywane do tworzenia testów jednostkowych w języku Java. Zaletą jest oddzielenie testów od kodu. Możliwość niezależnego uruchomienia i tworzenie raportów to tylko kilka ciekawych cech tego frameworka testującego. Służy do testowania kodu Java.

Protractor z kolei to narzędzie umożliwiające wykorzystanie pewnych schematów, zasad oraz skryptów w celu testowania aplikacji napisanej w Angular. Protractor uruchamiany jest na serwerze Selenium. Umożliwia testowanie, które symuluje wykonywanie pewnych czynności przez użytkownika, jak np. klikanie w element bądź wpisywanie tekstu.

#### 4.2.8. RabbitMQ

RabbitMQ jest darmowym zestawem narzędzi służący do asynchronicznego przesyłania komunikatów. Wykorzystuje Advanced Message Queuing Protocol(standard protokołu warstwy aplikacji pośredniczący w przekazywaniu komunikatów) i jest napisany w języku programowania Erlang. Biblioteki RabbitMQ są dostępne dla większości głównych języków.

# Projekt systemu

Celem rozdziału jest projekt i jego części z opisem wykorzystanych rozwiązań, które były konieczne w trakcie implementacji.

## Konfiguracja

Rozdział poświęcony wszelkim konfiguracjom wymaganym do poprawnego działania projektu systemu oraz aplikacji mobilnej systemu powiadomień ratowniczych.

### 5.1.1Konfiguracja maszyny serwera

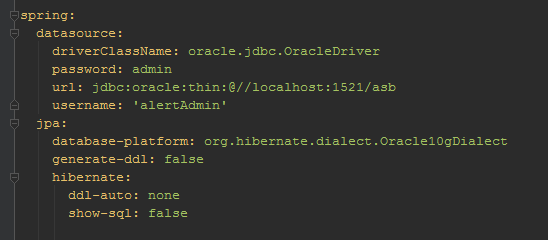
Poniżej opisane są kroki, które są wymagane do wykonania w celu uruchomienia systemu powiadomień ratowniczych.

System posiada mechanizm konfiguracji przez pliki z rozszerzeniem .yml. Istnieje jeden główny plik ‘application.yml’, w którym znajdują się wartości potrzebne w wersji deweloperskiej i produkcyjnej. Jeśli uruchomimy system z profilem ‘DEV’ załaduje się dodatkowo ‘application-dev.yml’, a jeśli z profilem ‘PROD’ to ‘application-prod.yml’. Tych profili nie można ze sobą łączyć, gdyż kontener nie będzie w stanie stworzyć różnych bean’ów o tej samej nazwie. Obsługa dwóch możliwości uruchomienia została wykonana przy pomocy sprofilowanych klas za pomocą adnotacji springowej ‘@Profile’.

#### Baza danych

Podczas tworzenia systemu jak i późniejszego rozszerzania system wykorzystuje bazę danych H2. Została wykorzystana wersja bazy danych zapisywana w pamięci RAM, co pozwala na łatwe i szybkie zmiany w strukturze modelu bazy. Wymaga instalacji bazy danych H2 na maszynie wykorzystywanej przy pisaniu oprogramowania systemu powiadomień.

Wersja produkcyjna korzysta z bazy danych Oracle. Stąd wymagane jest posiadanie takiej bazy w wersji nie mniejszej niż 11. Oraz zmodyfikowanie pliku ‘application-prod.yml’ znajdującego się w lokalizacji ‘resource/config’ wpisując w nim informacje o adresie bazy i danych dostępowych w sekcji ‘spring: datasource’.



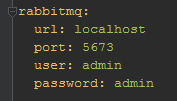
Fragment pliku application-prod.yml

#### Kontener aplikacji

System został przystosowany do uruchamiania na serwerze Apache Tomcat w wersji 8.0.24 i taki jest wymagany. Możliwe jest wykorzystanie innego kontenera jak JBOSS czy WebSphere wymaga to jednak zmian zależności bibliotek w głównym pliku .pom przystosowanych do niego, a w przypadku WebSphere prawdopodobnie wygenerowania pliku deskryptora. Jednak Tomcat jest na tyle wydajnym serwerem, że w zupełności wystarczy do wykonania celów projektu.

#### Kolejka

System do otrzymywania zgłoszeń wykorzystuje pośrednika komunikacji RabbitMQ, który nie musi być uruchomiony na maszynie, na której uruchamiany jest system. Wystarczy odpowiednia konfiguracja w pliku ‘application.yml’, zawierająca adres maszyny z RabbitMQ, port oraz dane autoryzujące.



Fragment pliku application.yml

## 5.2 Baza danych

Baza została skonstruowana jako relacyjny model. Obsługą zajmuje się framework Hibernate. Jest tylko wykorzystywana przez serwer, na którym uruchomiony jest system powiadomień ratowniczych.

Schemat bazy składa się z następujących tabel:

* User – zawiera informacje o użytkownikach mogących się zalogować do systemu, czyli operatorach pracujących przy obsłudze zgłoszeń oraz administratorów systemu. Zawiera takie dane jak:

- login,

-zakodowane hasło,

-imię,

-nazwisko,

-e-mail,

-aktywność konta,

-domyślny język,

-przypisane role.

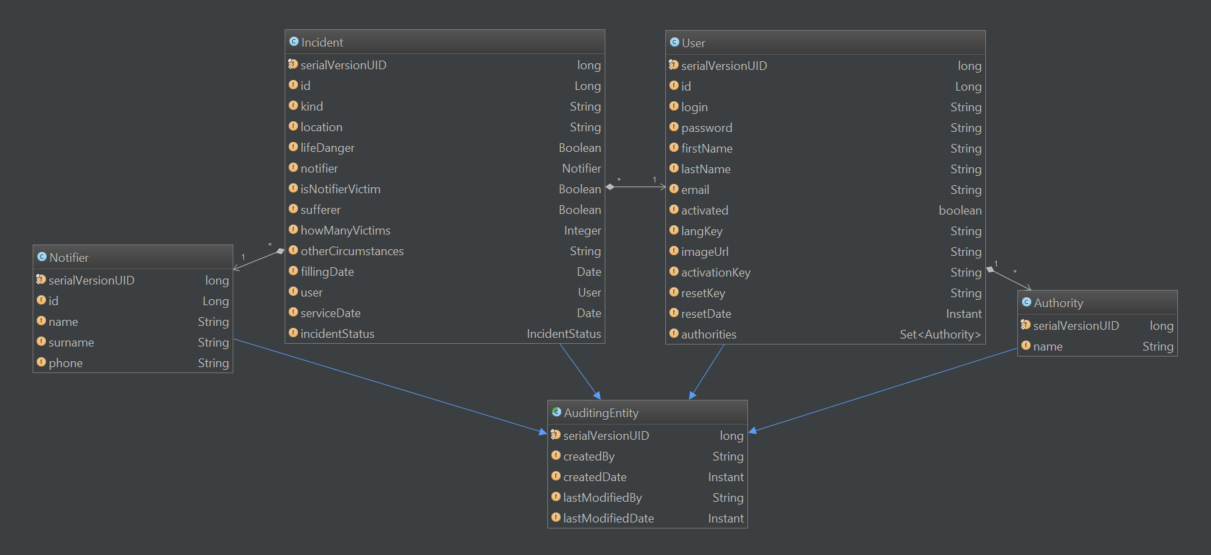
* Authority – jest to tabela stricte powiązana z tabelą User. Posiada role pracowników, które później mogą być przypisane do odpowiednich kont przez administratorów. Składa się z nazw tych ról.
* Notifier – w niej składowane są informacje o użytkownikach mobilnej aplikacji, którzy dokonali chociaż jednego zgłoszenia. Zawiera takie informacje jak:

-imię,

-nazwisko,

-numer telefonu.

* Incident – tabela przechowuje informacje o wszystkich zgłoszeniach znajdujących się w systemie. Składa się z informacji typu: kto zgłosił incydent, kto nad nim pracuje/pracował, najważniejsze informacje jak lokalizacja zgłoszenia, status zgłoszenia, czas obsługi i inne.
* AuditingEntity – encja abstrakcyjna, czyli nie posiadająca własnego odzwierciedlenia w tabelach bazy danych. Każda z tabel powyżej rozszerza tę tabelę, przez co każda zmiana na rekordzie którejkolwiek z tabel zaowocuje wpisaniem do wiersza informacji o tym kto i kiedy stworzył oraz kto i kiedy edytował te informacje. Została dodana z powodów bezpieczeństwa. Poniżej fragment kodu prezentujący ciało tej klasy.



Schemat bazy danych systemu



Klasa abstrakcyjna AuditingEntity

## 5.3 Pogrupowanie klas systemu powiadomień

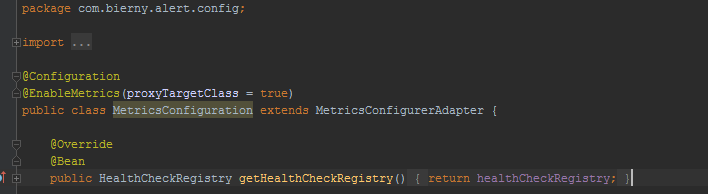
Ten podrozdział zajmie się opisem najważniejszych klas systemu i mobilnej aplikacji pogrupowanych w pakiety. Wszystkie klasy znajdują się w pakiecie rozpoczynającym się od ‘com.bierny.alert’.

#### 5.3.1. Logowanie

Jako, że zapisywanie informacji o zdarzeniach i problemach jest bardzo ważnym aspektem poprawności działania każdego systemu i jego późniejszych napraw oraz rozszerzania w pakiecie ‘logging’ znajdują się klasy, które obsługują tego rodzaju zapisy. Wszelka konfiguracja odnośnie logowania znajduje się tutaj. Istnieje jednak również możliwość konfiguracji poprzez odpowiednie wartości w plikach ‘applicaiton.yml’.

#### 5.3.2. Konfiguracja systemu

Wszelkie klasy odnośnie konfiguracji jak stałe konfiguracyjne, konfiguracja bezpieczeństwa, bazy danych, cache’a czy wszystkie uruchomieniowe zostały w pakiecie ‘config’ utworzone jako bean’y. Bean’y w przypadku framework’a Spring to obiekty tworzące szkielet aplikacji. Są to obiekty, które są instancjonowane, składane i zarządzane przez kontener Spring IoC. Tworzone są przez konfiguracje metadaty dostarczonej zazwyczaj w postaci tagów ‘<bean>’ w plikach xml, jednak w tym przypadku została użyta adnotacja springowa ‘@Bean’, a zdefiniowane zostały przy pomocy adnotacji ‘@Configuration’.



Fragment kodu wykorzystany przy metrykach systemu

W tym miejscu również jest wykorzystywana pamięć podręczna cache. Specyfika systemu wymusza bardzo częste sprawdzanie czy nie pojawiło się nowe zgłoszenie przez co takich zapytań jest bardzo dużo. Aby mniej obciążać bazę stworzony został taki cache. Wyczyszczenie cache wykonuje się przy każdym nadejściu zgłoszenia, poprzez uruchomienie zdarzenia , a pomiędzy, kiedy przeglądarka wysyła zapytanie o listę zgłoszeń to dane są brane z cache co przyspiesza działanie i odciąża bazę danych, która przy wielu zgłoszeniach może przybrać ogromną formę.

#### 5.3.3. Encje

Pakiet ‘domain’ zawiera wszystkie encje wykorzystywane w systemie. Encje to podstawowe pojęcia występujące w mapowaniu relacyjno-obiektowym. „Encja jest ‘rzeczą’, która może zostać wyraźnie zidentyfikowana. Konkretna osoba, firma lub zdarzenie są przykładami encji. Relacje są powiązaniami pomiędzy encjami. []„ Innymi słowy encje to odzwierciedlenia obiektów znajdujących się w świecie rzeczywistym, dzięki czemu można je zaprezentować w bazie danych.

To dodam do przypisu jako definicję  Chen, Peter Pin-Shan:

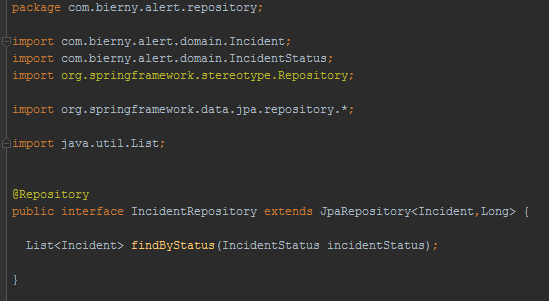
An entity is a “thing” which can be distinctly identified. A specific person, company, or event is an example of an entity. A relationship is an association among entities. For instance, “father-son” is a relationship between two CLperson” entities.’

#### 5.3.4. Repozytoria

Pakiet ‘repository’ to miejsce, gdzie składowane są klasy, które są wykorzystywane jako repozytoria JPA(Java Persistence Api). Służą one do zarządzania encjami i dzięki Spring stworzenie takiego bytu jest banalnie proste. Wystarczy użyć adnotacji ‘@Repository‘, a następnie rozszerzyć klasę o ‘JpaRepository’ interfejs. Po tym mamy już możliwość wykonania na encji podstawowych działań, między innymi takich jak:

* usuwanie danych,
* dodawanie danych,
* edycja danych,
* pobieranie danych,
* pobieranie danych ze stronicowaniem.

Przyspiesza to bardzo produkcję oprogramowania jednocześnie pozwalając pisać własne zapytania. Przykładem może być metoda ‘findByStatus’ w klasie ‘IncidentRepository’, która zwraca listę wszystkich zgłoszeń o podanym statusie.

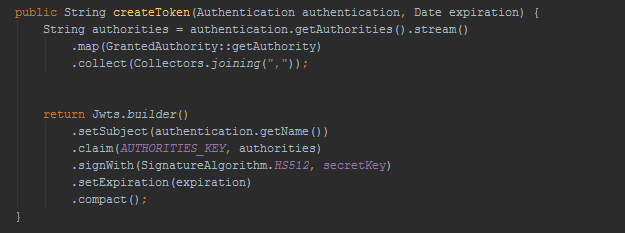


Klasa repozytorium encji Incident

#### 5.3.5. Bezpieczeństwo

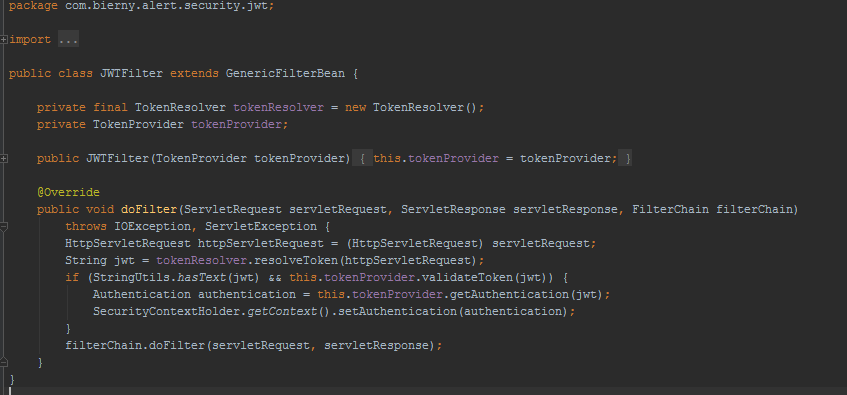
Jednym z wymagań systemu jest bezpieczeństwo. Za tę część po stronie Javy odpowiada pakiet ‘security’, w którym znajdują się klasy, które są mechanizmami obsługi logowań, rejestracji i sprawdzania dostępu danego użytkownika po jego roli.

W podpakiecie ‘jwt’ znajduje się obsługa tokenów dostępu, które zawierają poświadczenia bezpieczeństwa w ramach danej sesji i służą do identyfikacji. Token jest generowany przy procesie logowania do aplikacji i wysyłany do klienta. Token to zahaszowane informacje o autentykacji. Do wygenerowania takiego klucza wykorzystywana jest metoda z klasy ‘TokenProvider’.



Fragment klasy TokenProvider

Dzięki niemu system wie, że użytkownik ma prawo do danego zasobu. Każde zapytanie z przeglądarki jest wyłapywane przez filtr, w którym jest sprawdzana poprawność tokena.



Filtr sprawdzający poprawność tokena

#### 5.3.6. Serwisy

Kolejną grupą klas są serwisy. Serwisy to klasy udostępniające metody pełniące konkretne funkcje jak np. dodawanie nowego użytkownika czy akceptacja zgłoszenia. Są to klasy, które są wykorzystywane w całym systemie celem granulacji działań.

#### 5.3.7. REST

REST(Representational state transfer) czyli usługi sieciowe są sposobem zapewnienia interoperacyjności systemów w Internecie. Takie usługi umożliwiają żądanie o dostępy do danych zasobów i manipulowania nimi poprzez bezstanowe operacje. Krótko mówiąc są to wystawione usługi na zewnątrz systemu, do których każdy z dostępem do maszyny przez siec jest się w stanie skomunikować. Dlatego względy bezpieczeństwa wymagają również obsługi niepożądanych zapytań. Co zostało obsłużone w tym systemie przez politykę kluczy token.

REST bardzo szybko stał się standardem budowania serwisów sieciowych przez fakt, ze jest łatwy do budowania i łatwy w wykorzystywaniu. Ogromnym plusem jest działanie na protokole HTTP. Pozwala z łatwością dodać pamięć cache i zajmować się jego obsługą.

W systemie wszelkie kontrolery REST znajdują się w pakiecie ‘web.rest’.

#### 5.3.8. Główna klasa

Główną klasą inicjalizującą cały system jest klasa ‘AlertSystemApp’. Rozpoczyna ona działanie aplikacji i definicję bean’ów, zajmuje się uruchomienie nasłuchu na kolejkę RabbitMQ i wieloma innymi rzeczami wymaganymi do poprawnego wystartowania całego systemu.

## 5.4 Przeglądarkowa część systemu

Podrozdział ten zajmie się opisem składowych części przeglądarkowej systemu czyli tzw. „front-endu”. Zawartość strony wizualnej znajduje się w pakiecie ‘webapp’.

## 5.4.1 Składowe widoków

Na widoki(stronę wizualną) składają się przede wszystkim pliki HTML oraz JS. HTML zajmuje się ustawianiem komponentów na stronie i ich stylizacją, natomiast pliki JavaScript te dane do widoków dostarczają oraz zajmują się wieloma innymi rzeczami jak edycja danych po stronie przeglądarki, wywoływanie zdarzeń powiadamiających użytkownika czy ogólną obsługą działania komponentów HTML.

Jest kilka sposobów umieszczania takich plików w pakietach. Jednym z rozwiązań jest umieszczanie plików HTML w jednym miejscu, a plików JS w innym. Jest to jednak rozwiązanie, które przy powiększeniu systemu, powoduje trudności w odnajdywaniu elementów, które działają razem. Innym sposobem, użytym w tym projekcie, jest wydzielenie pakietów wg funkcjonalności lub nawet ekranów. Tak, że np. w pakiecie ‘app.admin.metrics’ znajdują się wszystkie składowe ekranu wyświetlającego poprawność działania systemu i jego obciążenie w części administracyjnej.

Wspólne komponenty nazywane w angularJS dyrektywami znajdują się z kolei w jednym miejscu, gdyż ich zastosowanie polega na wykorzystywaniu takich komponentów wielokrotnie w wielu miejscach.

## 5.4.2 Grafiki i style

Grafiki jak i pliki CSS również mają swoje miejsce, tak aby można było łatwo i szybko rozpocząć pracę nad ich zmianami czy ewentualnymi poprawkami.

## 5.4.3 Pliki językowe

System umożliwia zmianę języka przez co potrzebne jest miejsce przechowywania etykiet i wartości w kilku językach. W projekcie takie pliki przechowywane są w formacie JSON(JavaScript Object Notation).

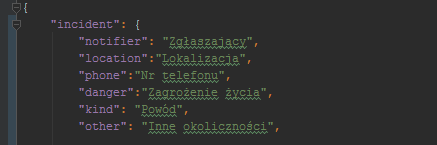
Działanie wersji w systemie polega na wykorzystaniu w widokach jedynie kluczy, a nie wartości, przez co można przypisać wartości do kluczy w plikach językowych.

Niech za przykład posłuży nagłówek tabeli na stronie odbioru zgłoszeń:



Fragment HTML widoku odbierania zgłoszeń

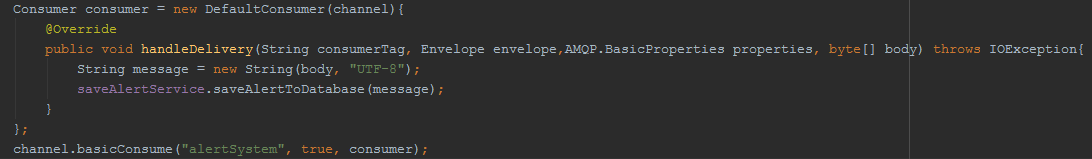
Dyrektywa ‘data-translate’ sprawdzi pliki językowe w poszukiwaniu odpowiednich kluczy, np ‘incident.location’. Jeśli znajdzie to wyświetli jej wartość. W przeciwnym wypadku pojawi się wartość domyślna wpisana bezpośrednio w plik HTML, czyli w tym przypadku ‘Location’. Jeśli jednak będzie ustawiony język polski, a w pakiecie ‘i18n.pl’ plik ‘incident.json’ będzie miał postać jak na fragmencie poniżej to na stronie wyświetli się wartość ‘Lokalizacja’.



Fragment pliku incident.json

## 5.5 Kolejka RabbitMQ

System wykorzystuje kolejkę, która znajduje się poza projektem i jest zainstalowana na maszynie. W celu sprawdzenia czy na kolejce nie pojawiła się nowa pozycja, należy stworzyć połączenie do tej kolejki i utworzyć metodę nasłuchującą, która podczas napływania powiadomień pobierze najnowszy i zapisze do bazy. Ta część wykonywana jest dopóki istnieje kolejny obiekt w kolejce. Jeśli już wszystkie zapisano to czeka na napływ kolejnych.



Mechanizm sprawdzający kolejkę RabbitMQ i zapisujący wszystkie zgłoszenia

## 5.6 Testy automatyczne

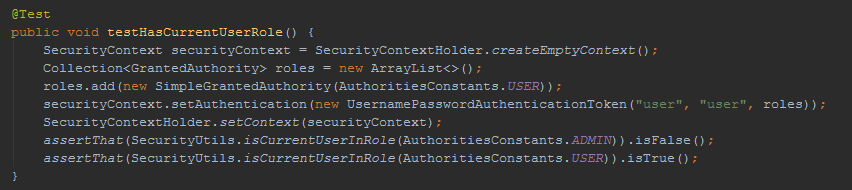
Istnieje technika tworzenia oprogramowania, która sprawdza się idealnie podczas jego produkcji. Jest to TDD, czyli Test-driven development. Polega ona na tym, ze pierwszym krokiem jest napisanie testu, który z racji nie zrobionej implementacji funkcjonalności przejdzie negatywnie. Następnie przechodzi się do tworzenia funkcjonalności i pracuje nad nią do momentu poprawnego przejścia testu jednostkowego. Taka metodyka sprawdza się znakomicie zapewniając tworzenie testów w czasie implementacji kodu. Taka metodyka została użyta przy niektórych aspektach projektu.

Testy zostały podzielone na dwie części:

* testy klas Java wykorzystujące JUnit,
* testy widoków przy użyciu Protractor.

Przykłady testów JUnit:

* test pierwszy:



Test sprawdzający czy użytkownik posiada poprawną rolę

Na potrzeby testu dodany został użytkownik o loginie i haśle ‘user’. Na początku testu tworzony jest kontekst i dodawana jest rola ‘USER’. Następnie wykonywana jest autentykacja na takiego użytkownika. Ostatnimi krokami jest sprawdzenie poprawności działania mechanizmów ustawiania i pobierania roli. Dokładniej mówiąc test sprawdza czy taki użytkownik posiada rolę ‘USER’, ale nie posiada roli ‘ADMIN’, która jest zarezerwowana dla administratorów systemu.

* test drugi:

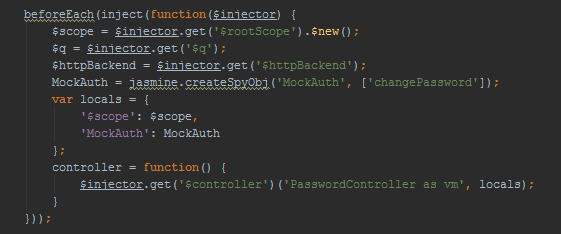


Test sprawdzający poprawność działania usługi restowej pobierającej jedno konkretne zgłoszenie

Test pobierania zdarzenia rozpoczynamy od dodania do bazy zgłoszenia testowego. Następnie przy użyciu ‘restIncidentMockMvc.perform’ wysyłamy zapytanie GET o pobranie konkretnego zdarzenia. Kolejnymi krokami jest sprawdzenie czy usługa zwróciła jakiekolwiek dane i czy są one w kodowaniu UTF-8. Na końcu sprawdzane są wartości jakie zwróciła usługa.

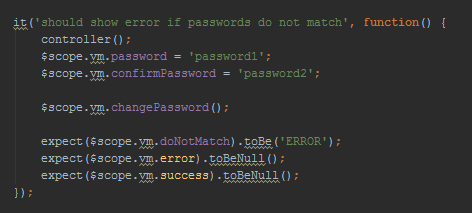
Przykład testów Protractor:

Podczas tworzenia testów Protractor sprawdzających poprawność działania części przeglądarkowej należy na początku stworzyć tak zwane ‘mocki’, czyli każdą zmienną i funkcję wykorzystaną w danej funkcjonalności musimy na skonstruować na potrzeby testu, gdyż chcemy, aby każdy test testował tylko jedną funkcjonalność, tak aby testy nie były od siebie zależne.



Funkcja beforeEach

Na początku tworzona jest funkcja ‘beforeEach’, która uruchomi się przed każdym testem w ramach danej funkcjonalności. W niej wpisane są wszystkie rzeczy wymagana do uruchomienia danego kontrolera, w tym przypadku ‘PasswordController’ wykorzystany przy zmianie hasła przez użytkownika. Tworzony jest tymczasowy zakres, zmienna wykorzystywana przy oczekiwaniu na wykonanie funkcji oraz wszelkie funkcje.



Test sprawdzający poprawność potwierdzającego hasła

Następnym krokiem jest napisanie testu. Pierwszy parametrem jest nazwa testu, która powinna opisywać co sprawdza test. Inicjowany jest kontroler zdefiniowany w funkcji ‘beforeEach’. W tym momencie ustawiane są elementy formularza z nowym hasłem na niezgadzające się ze sobą hasła. Wywoływana jest funkcja zmiany hasła. Na koniec sprawdzane jest czy wyświetlił się komunikat walidacyjny oraz sprawdzenie czy nie zostało wysłane zapytania o zmianę hasła, które mogłoby zwrócić błąd bądź przejść pozytywnie.

# Projekt aplikacji mobilnej

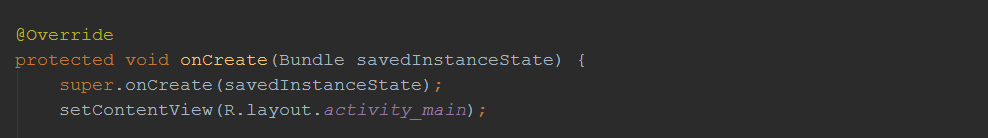
Celem rozdziału jest projekt i jego części z opisem wykorzystanych rozwiązań, które były konieczne w trakcie implementacji.

#### 6.1. Konfiguracja urządzenia mobilnego

Urządzenie mobilne nie wymaga żadnej konfiguracji poza zainstalowaniem aplikacji mobilnej systemu powiadomień. Jedynie użytkownik musi wyrazić zgodę na dostęp do modułu GPS urządzenia mobilnego przy pierwszym uruchomieniu. Aplikacja była uruchamiana na telefonie w wersji 7 systemu android oraz na emulatorze w wersjach 6, 8 oraz najnowszej wersji api udostępnionego przez twórców, które zostanie wykorzystane w nadchodzącej wersji systemu android.

#### 6.2. Klasy

Aplikacja mobilna składa się z aktywności. Każda aktywność posiada własną klasę i w ramach takiej klasy wykonywane są konkretne już funkcjonalności. Klasy są stricte połączone z widokami. Przejście pomiędzy aktywnościami polega na wywołaniu klasy, która na samym początku ustawia wybrany widok. Ustawianie jest konieczne w metodzie ‘onCreate’, która jest wymagana.

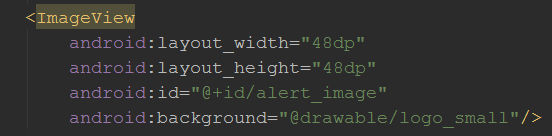


Fragment kodu klasy MainActivity

#### 6.3. Widoki

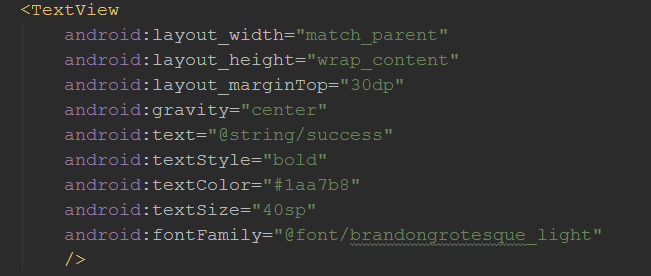
Widoki zdefiniowane są w plikach XML. Cała sztuka budowania widoku aplikacji mobilnej polega na utworzeniu układu, w którym umieszczamy kolejne podukłady, które mogą być pionowe bądź poziome i w nich ustawiane są komponenty. Każdy komponent posiada kilka atrybutów, dzięki którym można je dowolnie dostosowywać do naszych potrzeb. W projekcie użyto takich komponentów:

* ImageView – komponent wyświetlający obraz



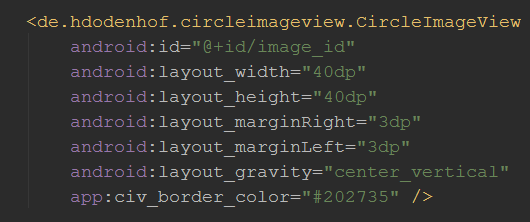
Komponent wyświetlający logo aplikacji

* TextView – ustawia sformatowany tekst



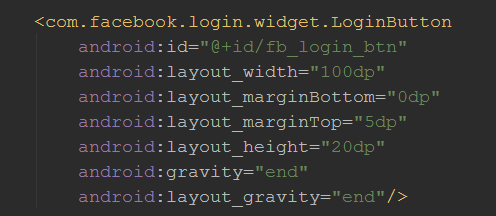
Komponent wyświetlający komunikat o pomyślnym przesłaniu zgłoszenia

* CircleImageView – pozwala wyświetlić obrazek w okręgu



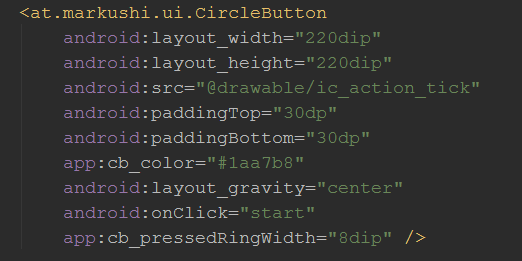
Komponent wyświetlający zdjęcie profilowe użytkownika

* facebook LoginButton – przycisk umożliwiający autoryzację przez Facebook



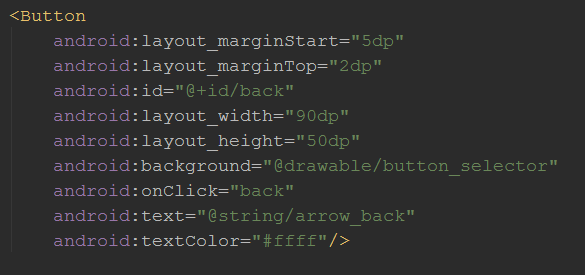
Komponent autoryzacji za pomocą Facebook

* CircleButton – przycisk w postaci okręgu



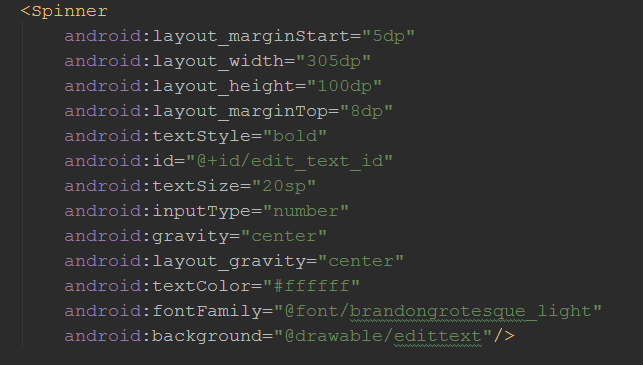
Komponent przycisku rozpoczęcia formularza zbierającego informacje zgłoszenia

* Button –generuje przycisk



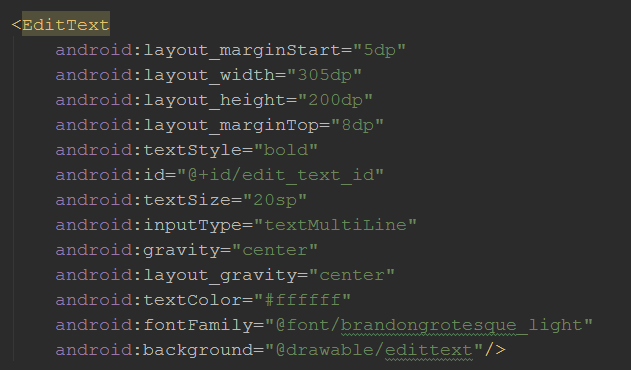
Komponent przycisku wstecz na formularzu

* Spinner – umożliwia dodanie wybieralnej listy rozwijalnej



Komponent wyboru rodzaju zdarzenia

* EditText – umożliwia dodanie pola tekstowego



Komponent dodawania wieloliniowego pola tekstowego przy opisie zdarzenia

W widokach została wykorzystana czcionka ‘brandongrotesque\_light’, która została wgrana do katalogu ‘font’.

Grafiki wykorzystane przy elementach jak przyciski czy obrazki umieszczone zostały w lokalizacji ‘drawable’ zgodnie z zasadami programowania w Android.

Wartości tekstowe nie powinny się znajdować w plikach definiujących widoki, a jedynie ich identyfikatory. Teksty definiowane są w plikach zewnętrznych, np. w pliku ‘strings.xml’, po to aby można było wykorzystać jedną wartość w wielu miejscach, co jest kluczowe przy ewentualnych poprawkach.

#### 6.4. Najważniejsze funkcjonalności

W tym podrozdziale opisane zostały kluczowe funkcjonalności dla projektu aplikacji mobilnej.

#### 6.4.1 Pobieranie lokalizacji GPS

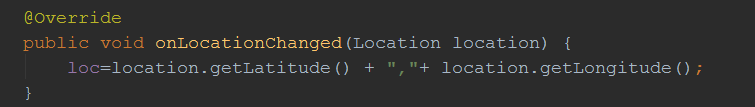
Aby pobrać lokalizację GPS należy wykonać kilka kroków. Pierwszym jest dodanie ‘LocationManager’, który jest elementem biblioteki każdej nowej wersji systemu Android. Następnie potrzebna jest metoda nasłuchująca zmiany w lokalizacji. Kiedy już posiadamy obie rzeczy to kolejnym krokiem jest wstrzyknięcie metody do managera i ustawienia dostawcy oraz co jaki czas i odległość ma się dana metoda wykonać.



Fragment mechanizmu pobierania lokalizacji GPS

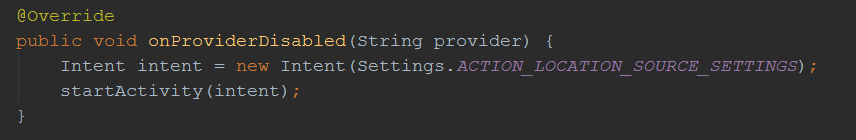
W metodzie nasłuchującej najważniejsze są dwie metody:

* onLocationChanged – która zostanie wywołana kiedy zmienimy lokalizację urządzenia mobilnego, tak aby zostały zaktualizowane dane lokalizacyjne,



Metoda ‘onLocationChanged’

* onProviderDisabled – metoda jest wywoływana, po to aby użytkownik musiał zgodzić się na wykorzystanie przez aplikację mobilną modułu GPS urządzenia. Wywołuje nową aktywność udzielenia zgody.



Metoda ‘onProviderDisabled’

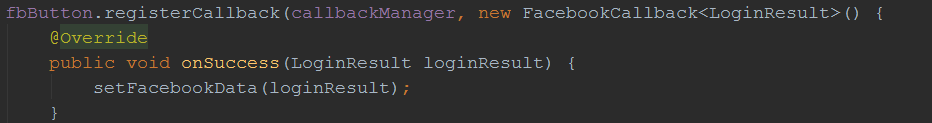
Należy pamiętać, że system Android wymaga sprecyzowania w aplikacji do jakich modułów urządzenia aplikacja chce korzystać. Takie uprawnienia definiowane są w głównym pliku ‘AndroidManifest.xml’. Do korzystania z lokalizacji GPS wymagane są uprawnienia ‘ACCESS\_FINE\_LOCATION’ oraz ‘ACCESS\_COARSE\_LOCATION’.



Uprawnienia GPS

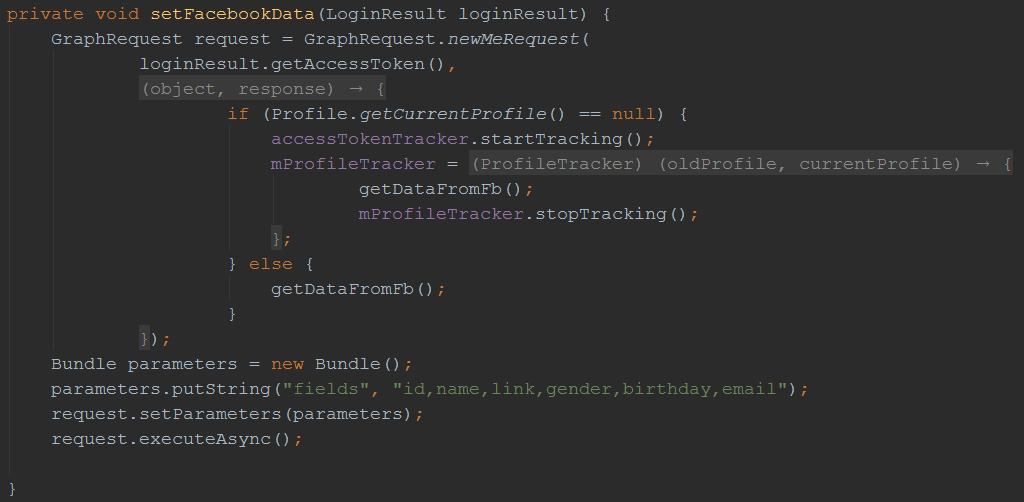
#### 6.4.2 Logowanie za pomocą Facebook

Facebook udostępnia biblioteki potrzebne do wykonania tej funkcjonalności. Po zaimportowaniu ich do projektu należy zainicjować ‘FacebookSDK’. Następnie przypisujemy do przycisku wywołanie zwrotne, które posiada metodę ‘onSuccess’, która uruchomi się po poprawnym zalogowaniu.



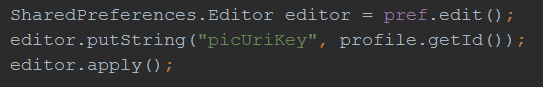
Rejestrowanie wywołania zwrotnego na przycisku logowania Facebook

Po poprawnej autentykacji przez serwis Facebook następuje pobranie danych ze zwracanego obiektu. Wykonuje się to w metodzie ‘setFacebookData’. Na początku tworzony jest graf z obiektu zwróconego przy logowaniu. Pobierana jest wartość profilu. Jeśli jej nie ma, a zazwyczaj dane profilu ładowane są jakiś czas po logowaniu, tak wynika ze specyfikacji bibliotek Facebook, potrzebujemy uruchomić kolejną metodę nasłuchującą ‘accessTokenTracker’, która poczeka na zmianę profilu i wtedy uruchomi uzupełnianie danych profilowych.



Metoda ‘setFacebookData’

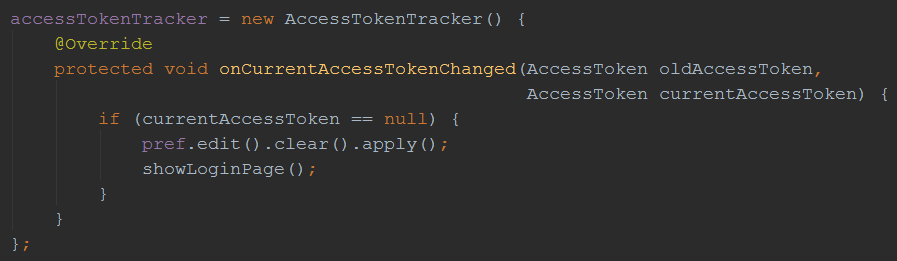
Wylogowanie działa z kolei na zasadzie ponownego uruchomienia funkcji nasłuchującej po poprawnym logowaniu i czekanie aż klucz token zmieni się ponownie na wartość pustą. Dane pobrane są przechowywane w pamięci ‘SharedPreferences’, tak aby po ponownym uruchomieniu aplikacji nie trzeba było się ponownie logować. Pamięć jest czyszczona przy wylogowaniu.



Dodawanie informacji o zdjęciu profilowym do pamięci dzielonej



Pobieranie informacji o imieniu z pamięci dzielonej przy tworzeniu użytkownika aplikacji



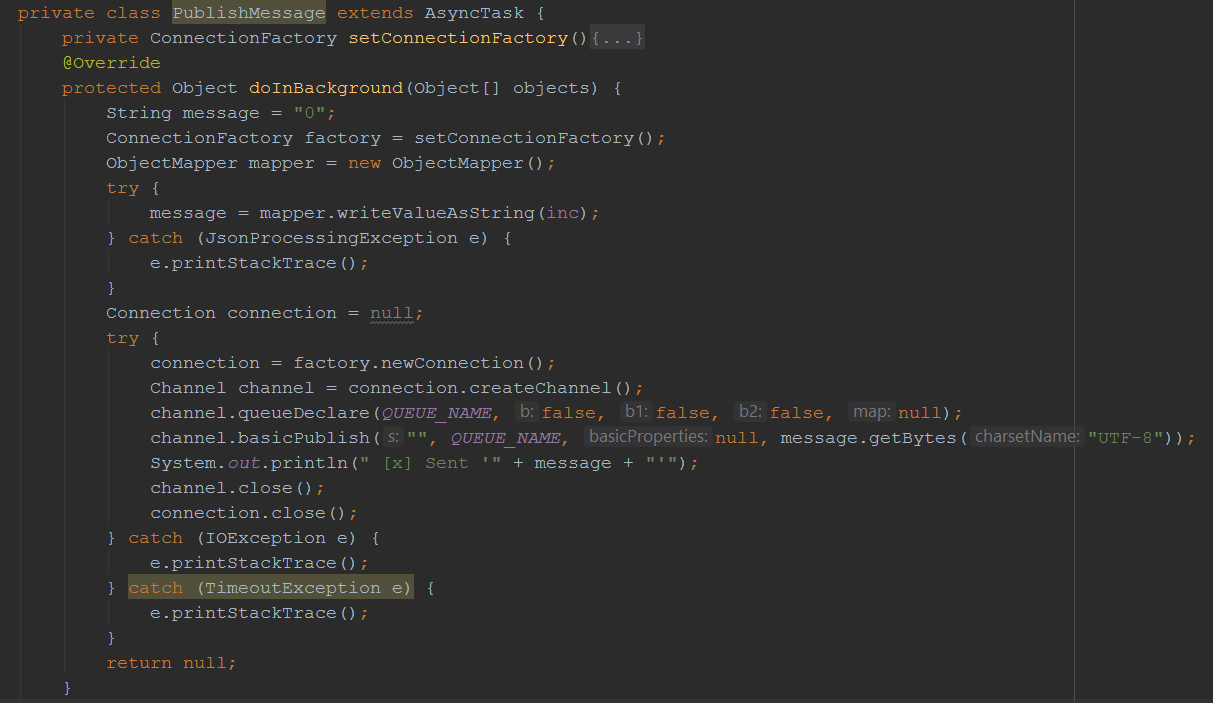
Metoda nasłuchująca wylogowanie i usuwająca dane z pamięci dzielonej oraz włączenie ekranu logowania.

#### 6.4.3 Wysyłanie zgłoszenia do kolejki RabbitMQ

Wysyłanie zgłoszenia do kolejki działa asynchronicznie stąd na potrzeby tej funkcjonalności należy stworzyć osobny wątek, który otrzyma obiekt i wyśle go na kolejkę.



Wywołanie nowego wątku wysyłania zgłoszenia



Kod wątku tworzącego połączenie z kolejką RabbitMQ i wysyłającego obiekt zgłoszenia