

Uniwersytet Mikołaja Kopernika w Toruniu  
Wydział Matematyki i Informatyki

**Dominik Cielicki**

nr albumu: 297001

Informatyka

Praca inżynierska

**Aplikacja mobilna wspierająca alergików**

Opiekun pracy dyplomowej  
Dr Aleksandra Boniewicz

# Zawartość

1. Wstęp.....	4
2. Przegląd istniejących rozwiązań .....	6
3. Wykorzystane technologie .....	9
3.1 API REST .....	9
3.1.1 Architektura API REST .....	9
3.1.2 Sposób działania interfejsów REST API .....	10
3.2 Technologie wykorzystane przy implementacji serwera REST .....	11
3.2.1 Java 8 .....	11
3.2.2 Git .....	13
3.2.3 Apache Tomcat .....	14
3.2.4 Apache Maven .....	14
3.2.5 Spring Framework .....	16
3.2.6 HSQLDB.....	16
3.2.7 Hibernate .....	16
3.2.8 JSON – biblioteka Gson.....	16
3.2.9 IntelliJ IDEA .....	18
3.3 Technologie wykorzystane przy implementacji aplikacji mobilnej .....	18
3.3.1 Android SKD .....	18
3.3.2 Android Studio.....	18
3.3.3 Biblioteka Volley .....	19
3.3.4 Gradle.....	19
4. Implementacja aplikacji .....	20
4.1 Implementacja serwera REST .....	20
4.1.1 Pliki konfiguracyjne.....	20
4.1.2 RestController .....	20
4.1.3 Klasy bazodanowe oraz compositekey .....	21
4.1.4 Repository .....	21
4.1.5 Service.....	22
4.2 Implementacja aplikacji mobilnej .....	23
4.2.1 Pliki konfiguracyjne.....	23
4.2.2 RecyclerView.....	24
4.2.3 Implementacja Chatu .....	24
4.2.4 Implementacja mapy .....	25
4.2.5 Implementacja logowania do Facebook.....	26

4.2.6 ViewModel .....	27
4.2.7 Tworzenie wyglądu w XML .....	28
4.2.8 Zmiana widoku w zakładce.....	29
5. Aplikacja AllergyApp .....	31
5.1 Instalacja i uruchomienie serwera .....	31
5.2 Uruchomienie aplikacji mobilnej .....	33
5.3 Wygląd i działanie zakładek aplikacji .....	34
5.3.1 Połączenie z kontem Facebook .....	34
5.3.2 Widok zakładki Alergeny .....	34
5.3.3 Widok zakładki Leki.....	35
5.3.4 Widok zakładki Chat.....	36
5.3.5 Widok zakładki Ustawienia .....	37
6. Podsumowanie .....	38
7. Bibliografia.....	39
8. Spis rysunków .....	40

# 1. Wstęp

Aplikacje mobilne w dzisiejszych czasach stale zyskują na popularności. Korzystanie z urządzeń stacjonarnych jest coraz częściej zastępowane urządzeniami przenośnymi, np. smartfonami. Z raportu z listopada 2021 roku „Badanie opinii publicznej w zakresie funkcjonowania rynku usług telekomunikacyjnych oraz preferencji konsumentów” przeprowadzonego przez Urząd Komunikacji Elektronicznej, wynika, że z usług telefonii komórkowej korzysta 96,9%, w tym ze smartfonów 78,0%. Badania te dowodzą, jak istotną rolę odgrywają urządzenia przenośne. Operacje takie jak słuchanie muzyki, rozrywka w postaci gier, robienie zdjęć, wysyłanie wiadomości czy oglądanie filmów, jest dostępne w każdej chwili.

Obserwacja zmieniających się trendów spowodowała, że podjąłem zagadnienie związane z aplikacją mobilną. Analizy poczynione na rynku potwierdziły, że nie istnieje aplikacja dla alergików, która będzie łączyć pewną społeczność i pełnić funkcję informacyjną o danym produkcie. Dlatego zdecydowałem się na stworzenie aplikacji AllergyApp.

Aplikacja AllergyApp udostępnia informacje o stężeniu alergenów w określonym wcześniej przez użytkownika terenie, za pomocą graficznej powłoki. Dodatkowo umożliwia użytkownikom dodawanie opinii oraz komentowanie leków dla alergików. Oprócz tego tworzy możliwości prowadzenia chatu.

Z danych statystycznych wynika, że nawet 40% Polaków cierpi na alergię, a do najczęstszych jej objawów zalicza się alergiczny nieżyt nosa, astma i pokrzywka. Te symptomy mogą być bardzo uciążliwe i znacznie obniżać dobrostan psychiczny i fizycznych. Nieleczona alergologia może doprowadzić do powstania poważnych chorób oddechowych. Dlatego alergicy powinni zapoznać się z kalendarzem pylenia i zadbać o to, aby w ich przeciwdziałać występującym w ich okolicy alergenów. W przypadku bardzo pomocna może okazać się aplikacja AllergyApp.

Wyniki badań ECAP (Epidemiologia Chorób Alergicznych w Polsce) wskazują, że główną przyczyną alergii są: roztocza pyłków, zarodniki grzybów, sierść i wydzieliny skórne zwierząt. Z tej listy szczególnie dokuczliwe są pyłki, ponieważ szybko się przemieszczają, a ich stężenie w powietrzu dynamicznie się zmienia. Dlatego ważne są aktualne prognozy dla alergików – pomocny jest zwłaszcza kalendarz pylenia dostępny we wspomnianej aplikacji AllergyApp.

Niniejsza praca ma charakter projektu. Poza stworzeniem i opisem aplikacji AllergyApp, rozważyłem w pracy wiele kwestii teoretycznych związanych z aplikacjami dla

alergików. Podjąłem takie zagadnienia, jak przegląd istniejących rozwiązań, wykorzystane technologie przy implementacji serwera REST i implementacji aplikacji mobilnej. W pracy sformułowałem też podsumowanie, zamieściłem bibliografię oraz dodałem spis rysunków.

## 2. Przegląd istniejących rozwiązań

Aplikacje dla alergików pozwalają monitorować stan zdrowia oraz stosować dawki leków adekwatnie do aktualnych potrzeb.

### 1) Aplikacja „Apsik”

„Apsik” to oprogramowanie przeznaczone dla osób zmagających się z alergią. Celem aplikacji jest udzielenie informacji osobom uczulonym na pyłki i różne rodzaje alergenów, które występują w danej okolicy. W trakcie tworzenia profilu podajemy Nick oraz wymieniamy rośliny, na których pyłki jesteśmy uczuleni.

Informację o stężeniu pyłków przedstawia się w interfejsie graficznym, ponadto aplikacja udostępnia dane o pogodzie i zanieczyszczeniu powietrza. Po podaniu lokalizacji na bieżąco jesteśmy informowani o poziomie pylenia pyłków w najbliższej okolicy. Aplikacja samoczynnie pobiera dane w czasie rzeczywistym z bieżących odczytów ze stacji pomiarowych Ośrodka Badania Alergenów Środowiskowych. W aplikacji dostępne są również opcje prowadzenia dzienniczka, który przykładowo może posłużyć do monitorowania efektywności działania leków (m.in. czy i w jaki sposób stosowane leki łagodzą skutki uczulenia) oraz czasomierza, który znajduje podobne zastosowanie.

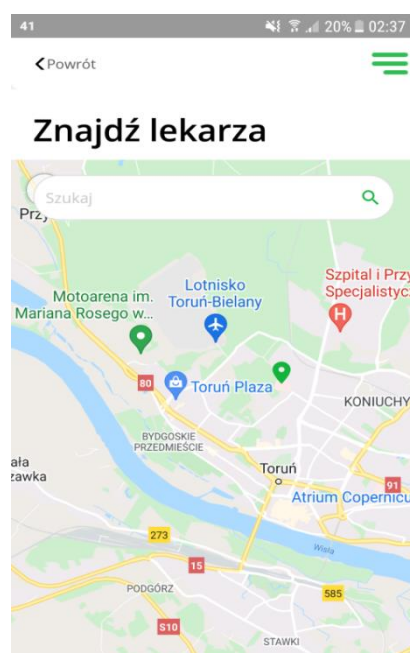
Aplikacja „Apsik” udostępnia informację o gabinetach specjalistów od alergii znajdujących się w pobliżu, w których za pomocą aplikacji można umówić się na wizytę. Warto wspomnieć o funkcji pozwalającej na konwertowanie dzienniczka do pliku PDF, który może się przydać podczas wizyty u specjalisty.



Rysunek 1 Mapa pyleń, "Apsik"



Rysunek 2 Strona główna, "Apsik"

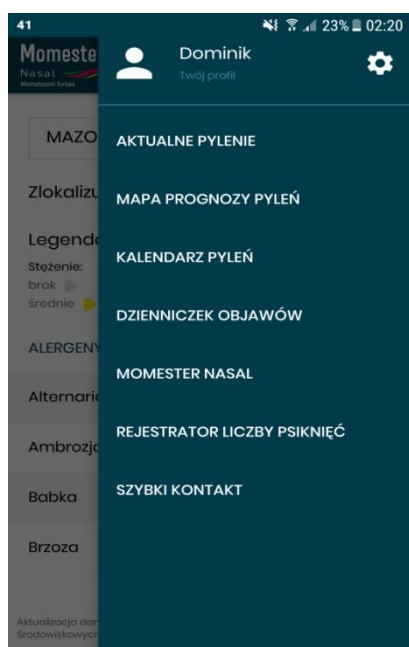


Rysunek 3 Mapa gabinetów, "Apsik"

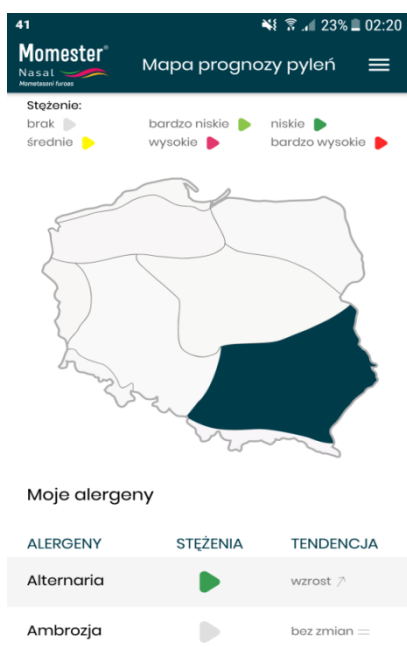
## 2) Aplikacja „Momester Nasal”

Podobne funkcje do „Apsik” realizuje również inna aplikacja dla alergików - Momester Nasal. Można znaleźć w niej wiele przydatnych informacji na temat stężenia alergenów w wybranej lokalizacji, kalendarz pyleń dla całego kraju, jak również możliwość oceny aktualnego samopoczucia oraz występujących objawów alergicznych. Walorem tej aplikacji jest opcja umożliwiająca ewidencjonowanie dawek zastosowanego leku. Korzystając z inhalatora czy innych farmaceutyków można na bieżąco kontrolować ilość przyjmowanego leku, a jednocześnie przeanalizować jego wpływ na samopoczucie.

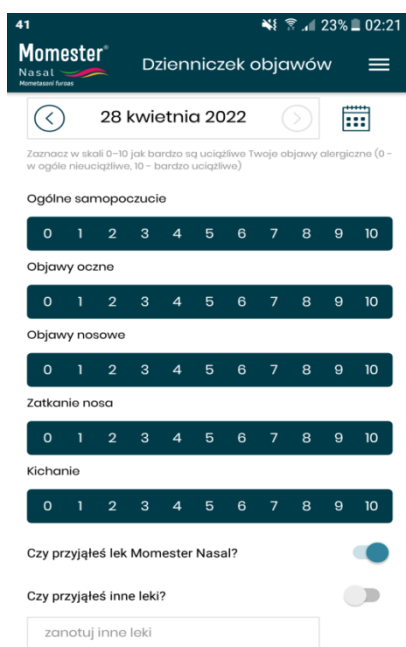
Aplikacja Momester Nasal umożliwia szybki kontakt ze specjalistą w przypadku sytuacji zagrażającej zdrowiu czy życiu.



Rysunek 4 Menu, "Momester Nasal"



Rysunek 5 Mapa pyleń, "Momester Nasal"

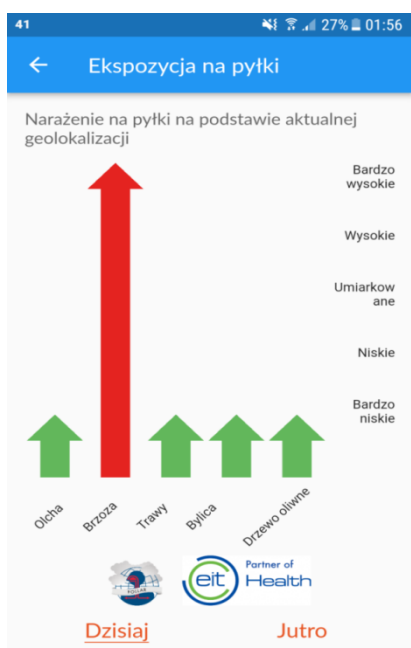


Rysunek 6 Dzienniczek objawów, "Momester Nasal"

## 3) Aplikacja „MASK-air”

MASK-air to dość specyficzna aplikacja dla alergików, służy ona do monitorowania alergicznego nieżytu nosa, tj. kataru siennego. Pozwala ona na monitorowanie stanu zdrowia, tak aby zastosować odpowiednie działania lecznicze i skutecznie zapobiegać różnym schorzeniom alergicznym. np. astmie. Aplikacja pozwala generować raporty, które pozwalają specjalistom na ich podstawie uzyskać odpowiednią diagnozę pacjenta.

Dzięki przyjęciu właściwego środka leczniczego można znacznie poprawić własny komfort życia i dobrostan psychiczny. Warto dodać w tym kontekście, że aplikacje dla alergików dostarczające aktualne informacje z Ośrodka Badania Alergenów Środowiskowych, pozwalają na przygotowanie się na zmiany środowiskowe, w wielu sytuacjach, na przykład w podróży. Jakkolwiek aplikacje dla alergików nie rozwiążą problemów zdrowotnych, to w znacznym stopniu pomogą im lepiej zadbać o stan swojego zdrowia.



Rysunek 7 Wykres stężenia pyłków, "MASK-air"



Rysunek 8 Panel główny, "MASK-air"

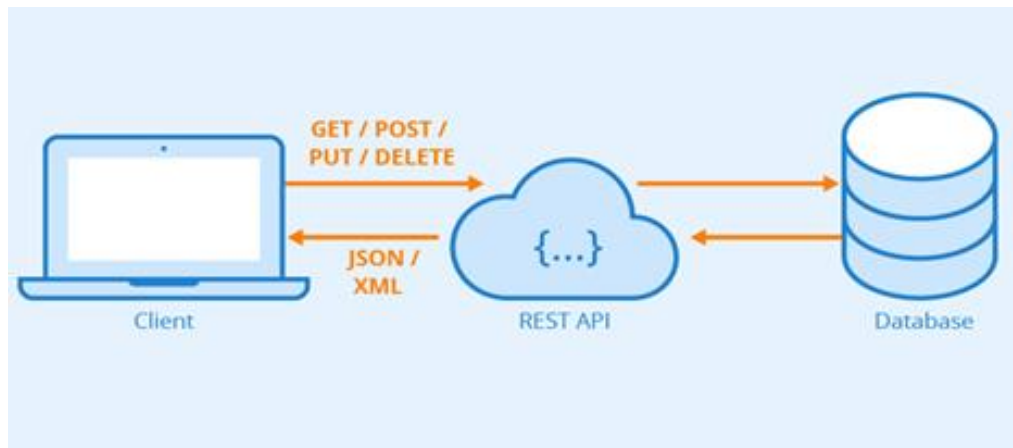


## 3. Wykorzystane technologie

### 3.1 API REST

W tym rozdziale zostaną przedstawione i opisane narzędzia, które zostały użyte podczas tworzenia aplikacji.

#### 3.1.1 Architektura API REST



Rysunek 9 Architektura API REST

Komunikacja aplikacji AllergyApp odbywa się przy użyciu interfejsu API REST. Sposób implementacji połączenia, na jaki się zdecydowałem, jest obecnie najbardziej powszechnym sposobem integracji komponentów w aplikacjach.

Interfejs API (Application Programming Interface) to zestaw reguł definiujący komunikację systemów komputerowych, aplikacji lub urządzeń, natomiast specyfikacja REST (Representational State Transfer) to styl architektury oprogramowania, opierający się na wcześniej określonych regułach, pozwalający na zdefiniowanie formatu przesyłanych danych. Architektura Rest została zaprojektowana przez Roya Fieldinga w 2000 roku. Zaletą takiego podejścia jest fakt, że API REST może być obsługiwane przez każdy format danych, a także implementowane w każdym języku programowania.

Architektura API REST powinna spełniać następujące warunki:

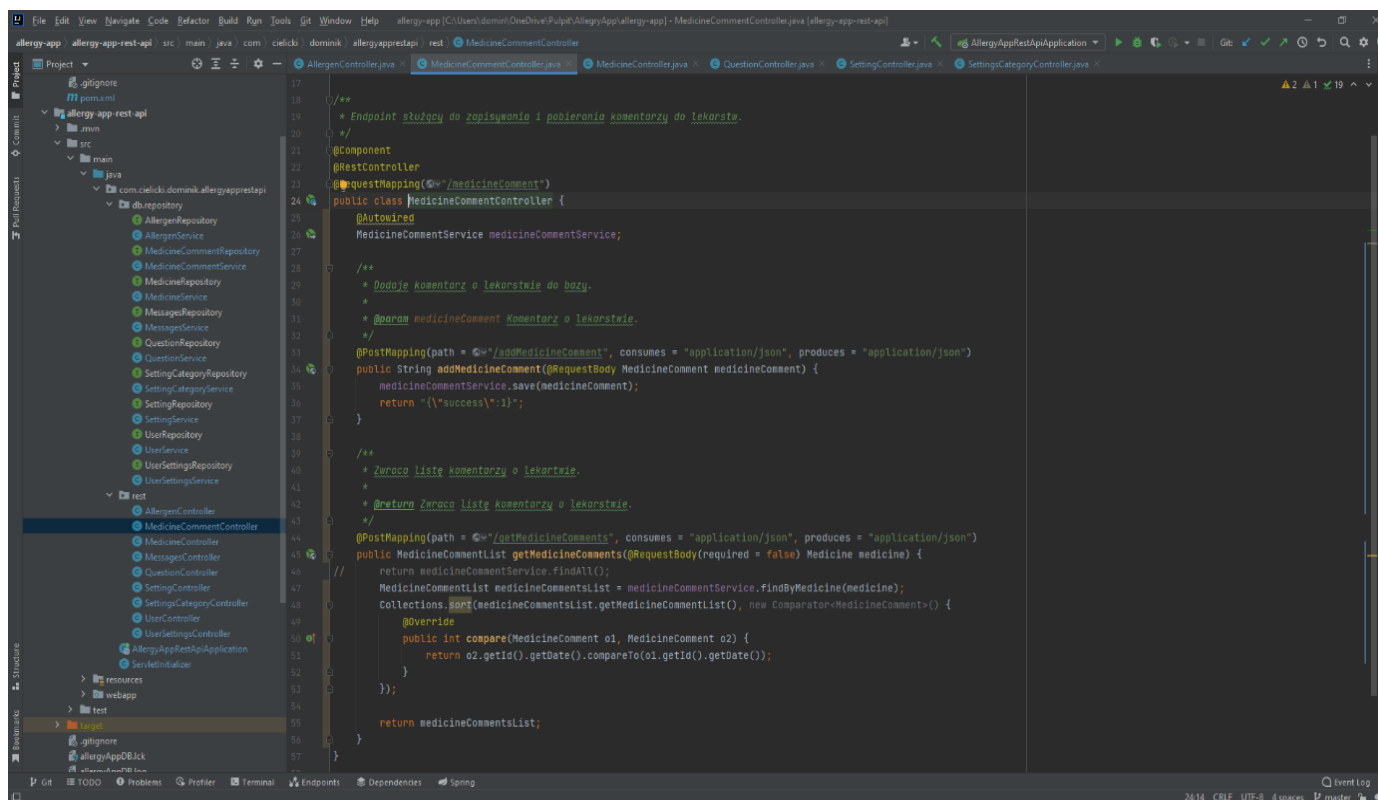
- jednolity interfejs – oznacza, że serwer musi udostępnić usługę API;
- rozdzielenie klienta i serwera - oznacza, że klient otrzymuje dostęp za pomocą udostępnionej przez serwer usługi API;
- bezstanowość – oznacza, że serwer nie może posiadać zaimplementowanych mechanizmów przetrzymujących dane klienta, które byłyby wymagane do poprawnego działania aplikacji;
- możliwość buforowania – oznacza, że żądania, które trafiają do serwera powinny być zapisywane w kolejce;
- warstwowa architektura systemu – oznacza, że aplikacja kliencka nie powinna posiadać danych o usługach i serwisach, z jakich serwer może korzystać.

### **3.1.2 Sposób działania interfejsów REST API**

Interfejsy API REST komunikują się za pośrednictwem żądań HTTP (Hypertext Transfer Protocol), aby wykonać standardowe funkcje bazy danych. Istnieje dziewięć metod HTTP:

- GET
- POST
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE
- PATCH
- HEAD

Zwykle do tworzenia mniej rozbudowanych aplikacji w pełni wystarczają tylko pierwsze cztery metody wymienione powyżej. Metoda GET służy do pobrania rekordu, POST do żądania w celu utworzenia rekordu, PUT do aktualizacji rekordu, zaś DELETE do usunięcia rekordu.



Rysunek 10 Przykład PostMapping

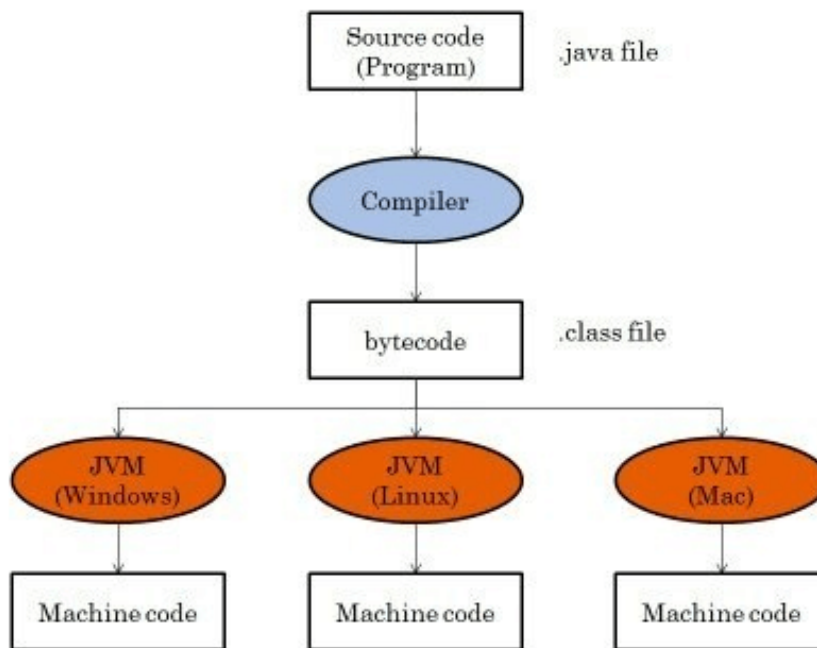
## 3.2 Technologie wykorzystane przy implementacji serwera REST

### 3.2.1 Java 8

Językiem programowania, którym posłużyłem się do implementacji serwera jest Java<sup>1</sup> w wersji 8. Istnieje powszechna opinia wśród programistów, że w Javie programowanie jest szybkie i proste. Powodem takiego przekonania jest to, że posiada on obszerną ilość gotowych bibliotek, które ułatwiają i przyspieszają tworzenie aplikacji. Java została opracowana i wydana w 1995 roku pod kierunkiem Jamesa Goslinga z firmy Sun

<sup>1</sup> <https://www.java.com/pl/>

Microsystems. Java została zaprojektowana w stylu składni języka Smalltalk oraz C++. Do dnia dzisiejszego Java jest jednym z najbardziej rozpowszechnionych technologii. Swoją popularność zyskała dzięki możliwości uruchomienia na dowolnym systemie operacyjnym. Stało się to możliwe poprzez wykorzystanie maszyny wirtualnej Java (Java Virtual Machine – JVM). Kompilator Java z plików .java tworzy pliki .class, które zawierają tzw. bytecode, który można uruchomić na JVM.



Rysunek 11 Java Virtual Machine (JVM)

Java jest językiem zorientowanym obiektowo, co w pewien sposób ułatwia tworzenie kodu, który można wykorzystać ponownie. Do głównych koncepcji wymienionego języka programowania należą:

- obiektość;
- dziedziczenie;
- niezależność od architektury;
- sieciowość i obsługa programowania rozproszonego;
- niezawodność i bezpieczeństwo (np. system wyjątków).

### 3.2.2 Git

W trakcie pracy nad projektem korzystałem z rozproszonego systemu kontroli wersji – Git<sup>2</sup>, który został stworzony przez Linusa Torvalds’a w 2005 roku i powstał z myślą o wspomaganiu rozwoju jądra Linux. System kontroli wersji posłużył mi do śledzenia zmian oraz przywracania poprzednich wersji aplikacji.

```
domin@DESKTOP-7DE6BVF MINGW64 ~/OneDrive/Pulpit/AllegryApp/allergy-app (master)
$ git log
commit a10a8b239c4416c7920a603aca3d5de1728990fd (HEAD -> master, origin/master, origin/HEAD)
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Sat Apr 23 18:19:34 2022 +0200

    Poprawki wysyłania wiadomości

commit 8d20030fea6bde5c8e70678d59c5c04aed5ff89
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Thu Apr 21 03:15:32 2022 +0200

    Dodanie komentarzy i usunięcie zbędnych importów

commit 74b5b9a0694c17bfc9a08f88d610770e4a0d7117
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Thu Apr 21 01:42:42 2022 +0200

    Dodanie logowania do facebooka oraz dodanie obsługi chatu globalnego

commit 662e0e6757def1935a28e37f7f7cf04f39bd2692
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Thu Apr 21 01:42:00 2022 +0200

    Zabezpieczenie servletu za pomocą klucza oraz uzupełnienie dokumentacji

commit ba58b98727c488d05ed255d03ed51709b6343047
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Sun Apr 10 17:55:11 2022 +0200

    Poprawki w REST Api i rozwój aplikacji

commit e323cd0db0cae99abb31aa3c91a3f24e8d0f9c2e
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Sun Mar 13 02:50:29 2022 +0100

    Poprawki Rest API i implementacja logiki zakładki leki

commit d5b0fcea054eb9aef920c893f3a4351ec22f2031e
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Sun Jan 23 01:18:41 2022 +0100

    Dodanie komentarzy i drobne zmiany

commit 5a44669470e598580955cdf899edcf371f5a66ee
Author: Dominik Cielicki <dominik275275275@wp.pl>
Date: Mon Jan 17 00:12:06 2022 +0100

    Dodanie komunikacji z RestApi

commit d90893a31260b4d76ab02a3dc15640938a7a18c8
Author: Dominik Cielicki <dominik275275275@wp.pl>
Date: Mon Jan 17 00:11:21 2022 +0100

    Dodanie klasy do dodawania testowych danych

commit 30ee0314076856bb6bfcf01387b573b87a59c5ea
Author: Dominik Cielicki <dominik275275275@wp.pl>
Date: Sun Jan 9 22:01:43 2022 +0100

    Podzielenie projektu REST Api i dodanie projektu aplikacji android

commit 3a0824d3cd8f063a8ba84c0b08ad9ad643c75ad1
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Sun Dec 19 00:42:33 2021 +0100

    Stworzenie pozostałych endpointów

commit 8675d2e77d73735e270503260fa477892809a73c
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Sun Dec 5 23:37:23 2021 +0100

    Poprawki w klasie MedicineCommentId

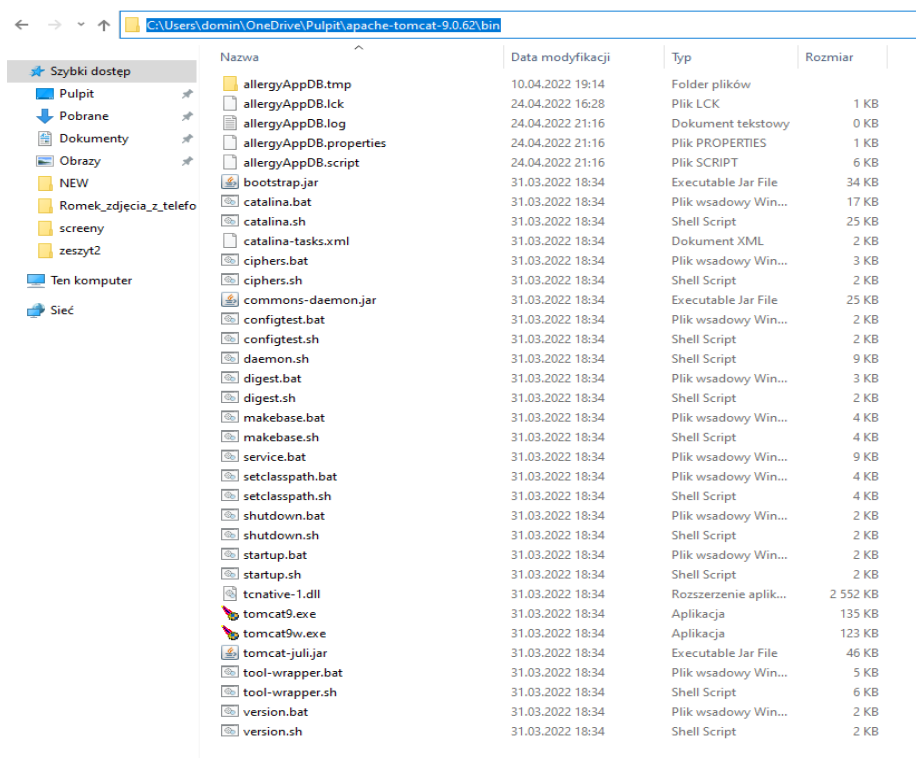
commit cdb466ef47100b868a53d1aea3b973a4e6cf48cb
Author: Dominik Cielicki <dominikcielicki1999r@wp.pl>
Date: Sun Dec 5 22:19:06 2021 +0100
```

Rysunek 12 Git log

<sup>2</sup> <https://git-scm.com/>

### 3.2.3 Apache Tomcat

Serwer aplikacji został uruchomiony na kontenerze aplikacji sieciowych Apache Tomcat<sup>3</sup>. Serwer Tomcat pozwala na uruchamianie aplikacji internetowych w technologii Java Servlets. Jest darmowym narzędziem i jednym z najpopularniejszych kontenerów sieciowych. Dzięki szerokiemu zastosowaniu używany jest nawet w największych korporacjach.



Nazwa	Data modyfikacji	Typ	Rozmiar
allergyAppDB.tmp	10.04.2022 19:14	Folder plików	
allergyAppDB.lck	24.04.2022 16:28	Plik LCK	1 KB
allergyAppDB.log	24.04.2022 21:16	Dokument tekstowy	0 KB
allergyAppDB.properties	24.04.2022 21:16	Plik PROPERTIES	1 KB
allergyAppDB.script	24.04.2022 21:16	Plik SCRIPT	6 KB
bootstrap.jar	31.03.2022 18:34	Executable Jar File	34 KB
catalina.bat	31.03.2022 18:34	Plik wsadowy Win...	17 KB
catalina.sh	31.03.2022 18:34	Shell Script	25 KB
catalina-tasks.xml	31.03.2022 18:34	Dokument XML	2 KB
ciphers.bat	31.03.2022 18:34	Plik wsadowy Win...	3 KB
ciphers.sh	31.03.2022 18:34	Shell Script	2 KB
commons-daemon.jar	31.03.2022 18:34	Executable Jar File	25 KB
configtest.bat	31.03.2022 18:34	Plik wsadowy Win...	2 KB
configtest.sh	31.03.2022 18:34	Shell Script	2 KB
daemon.sh	31.03.2022 18:34	Shell Script	9 KB
digest.bat	31.03.2022 18:34	Plik wsadowy Win...	3 KB
digest.sh	31.03.2022 18:34	Shell Script	2 KB
makebase.bat	31.03.2022 18:34	Plik wsadowy Win...	4 KB
makebase.sh	31.03.2022 18:34	Shell Script	4 KB
service.bat	31.03.2022 18:34	Plik wsadowy Win...	9 KB
setclasspath.bat	31.03.2022 18:34	Plik wsadowy Win...	4 KB
setclasspath.sh	31.03.2022 18:34	Shell Script	4 KB
shutdown.bat	31.03.2022 18:34	Plik wsadowy Win...	2 KB
shutdown.sh	31.03.2022 18:34	Shell Script	2 KB
startup.bat	31.03.2022 18:34	Plik wsadowy Win...	2 KB
startup.sh	31.03.2022 18:34	Shell Script	2 KB
tcnative-1.dll	31.03.2022 18:34	Rozszerzenie aplik...	2 552 KB
tomcat9.exe	31.03.2022 18:34	Aplikacja	135 KB
tomcat9w.exe	31.03.2022 18:34	Aplikacja	123 KB
tomcat-juli.jar	31.03.2022 18:34	Executable Jar File	46 KB
tool-wrapper.bat	31.03.2022 18:34	Plik wsadowy Win...	5 KB
tool-wrapper.sh	31.03.2022 18:34	Shell Script	6 KB
version.bat	31.03.2022 18:34	Plik wsadowy Win...	2 KB
version.sh	31.03.2022 18:34	Shell Script	2 KB

Rysunek 13 Apache-tomcat, folder bin

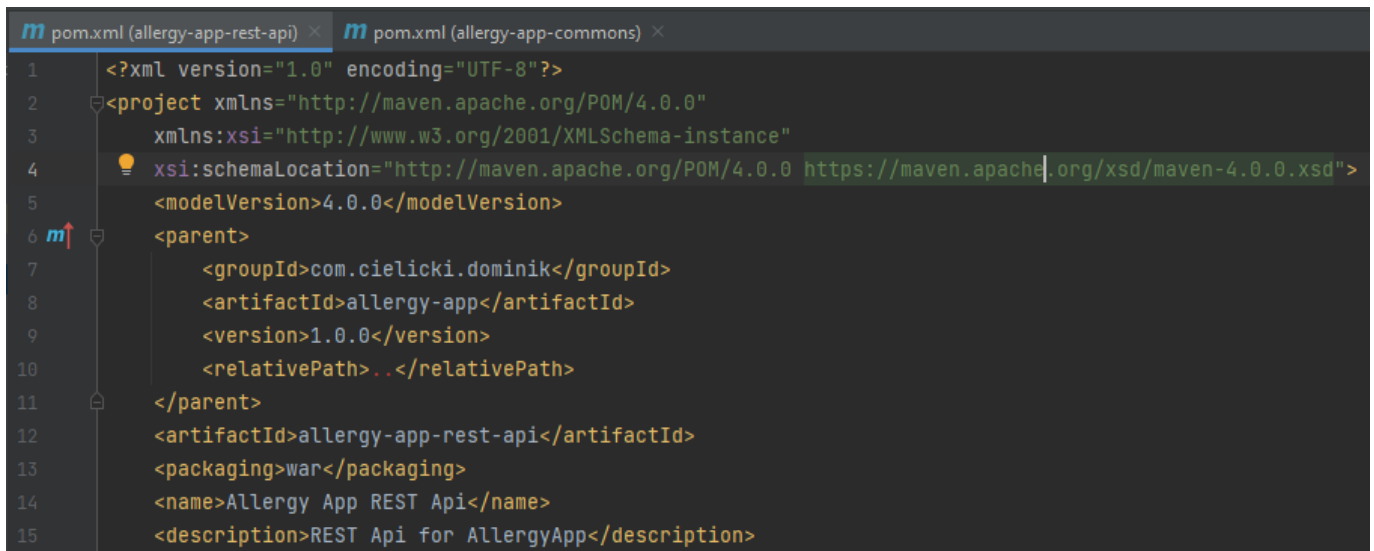
### 3.2.4 Apache Maven

Maven<sup>4</sup> jest narzędziem do budowania i zarządzania projektem. Było ono stworzone w 2002 roku przez Jasona van Zyla. W początkowej fazie był podprojektem Apache Turbine. Dopiero w lipcu 2004 roku została wydana pierwsza wersja jako jeden z samodzielnych projektów Apache Software Foundation.

<sup>3</sup> <https://tomcat.apache.org/>

<sup>4</sup> <https://maven.apache.org/>

Konfiguracja zapisywana jest w plikach pom.xml



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6     <parent>
7         <groupId>com.cielicki.dominik</groupId>
8         <artifactId>allergy-app</artifactId>
9         <version>1.0.0</version>
10        <relativePath>../</relativePath>
11    </parent>
12    <artifactId>allergy-app-rest-api</artifactId>
13    <packaging>war</packaging>
14    <name>Allergy App REST Api</name>
15    <description>REST Api for AllergyApp</description>
```

Rysunek 14 pom.xml

Do głównych cykli życia projektu maven'owego możemy zaliczyć:

- **clean** – czyszczenie pliku stworzonego podczas budowania;
- **validate** – sprawdzenie poprawności konfiguracji;
- **compile** – kompilacja kodu źródłowego;
- **test** – uruchomienie testów jednostkowych;
- **package** – spakowanie paczki jar lub war, w zależności od konfiguracji;
- **integration-test** – uruchomienie testów integracyjnych na środowisku testowym;
- **verify** – sprawdzenie poprawności zbudowanej paczki;
- **install** – przeniesienie paczki do repozytorium lokalnym maven'a;
- **deploy** – wysłanie paczki do repozytorium zdalnego.

W projekcie wykorzystałem Maven głównie do zarządzania zależnościami projektu oraz do budowania projektu. Moduł projektu allergy-app-commons wykorzystywany jest również w aplikacji mobilnej. Dzięki instalacji modułów w lokalnym repozytorium, projekt aplikacji mobilnej może z niego korzystać.

### **3.2.5 Spring Framework**

Pierwsza wersja Spring Framework została wydana w październiku 2002 roku przez Roda Johnsona. Jest frameworkiem aplikacyjnym, który w znaczący sposób przyspiesza tworzenie aplikacji webowych w języku Java. Zapewnia on łatwo konfigurowalne środowisko. Spring korzysta ze wstrzykiwania zależności (dependency injection) na podstawie adnotacji java (annotations), które eliminują potrzebę inicjalizacji obiektów. Instancje odpowiednich klas są dopasowywane na podstawie adnotacji i typu klasy. Framework umożliwia również tworzenie punktów dostępowych dla serwera REST, również na podstawie adnotacji, co zaprezentuję w rozdziale poświęconym implementacji serwera.

### **3.2.6 HSQLDB**

HyperSQL powstał w 2001 roku dzięki HSQL Development Group. Jest to silnik bazodanowy oparty na języku Java. Baza ta jest szeroko wykorzystywana w procesie tworzenia, testowania i wydawania aplikacji. Jest łatwa w wykorzystaniu i pracuje w pamięci komputera, uruchamiana z pliku .jar. Wykorzystałem ją ze względu na prostotę konfiguracji oraz integracji z frameworkiem Spring.

### **3.2.7 Hibernate**

Spring framework do manipulacji danych wykorzystuje Framework Hibernate. Jest to platforma programistyczna umożliwiająca mapowanie klas Java na obiekty bazodanowe, a typy danych Java na odpowiednie rodzaje danych w bazie danych. Hibernate, podobnie jak Spring, również wykorzystuje adnotacje. Umożliwia on definiowanie relacji między tabelami, co zostanie zaprezentowane w rozdziale dotyczącym implementacji serwera.

### **3.2.8 JSON – biblioteka Gson**

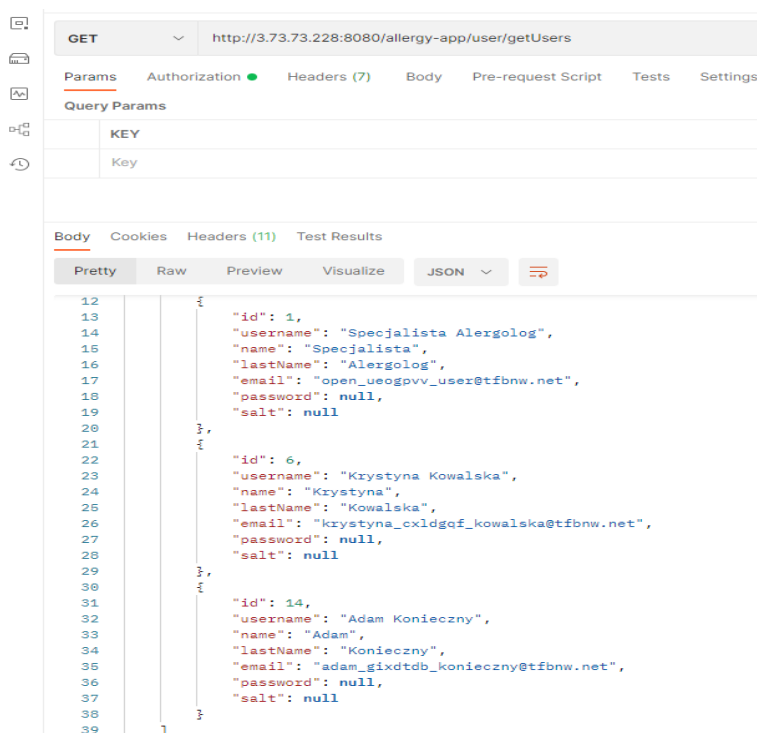
Informacje o stanie zasobu, służące do przechowywania i transportu danych mogą być przekazywane w formatach JSON (Java Script Object Notation), HTML, XLT, Python, PHP lub też jako jawny tekst. Formatem, którym się posłużyłem jest JSON. Uznawany jest za najbardziej popularny dla aplikacji z użyciem języka Java i Spring Boota.



```
{
  "oceny": [
    {"przedmiot": "matematyka", "ocena": 5},
    {"przedmiot": "przyroda", "ocena": 3},
    {"przedmiot": "religia", "ocena": 6}
  ]
}
```

Jak widać na przykładzie powyżej, JSON może zawierać obiekty (zamknięte w nawiasy klamrowe), listy (zamknięte w kwadratowych nawiasach) oraz pary w formacie nazwa: wartość.

Zarówno w projekcie serwera, jak i aplikacji mobilnej do przetwarzania JSON'ów korzystam z biblioteki od firmy Google - Gson. Przy testowaniu przekazywania danych wymienionego formatu korzystałem z narzędzia Postman. Narzędzie Postmana posłużyło mi do testowania backendu, przy sprawdzaniu poprawności działania adresów.



Rysunek 15 Format JSON, Postman

### **3.2.9 IntelliJ IDEA**

Zintegrowanym środowiskiem programistycznym, na którym pracowałem, był IntelliJ IDEA<sup>5</sup>. Jest to intuicyjne narzędzie wspierające Git oraz Maven, oprócz tego w znaczącym stopniu przyspiesza i ułatwia pracę poprzez formatowanie kodu, podpowiadanie oraz sprawdzanie poprawności składni języka programowania.

## **3.3 Technologie wykorzystane przy implementacji aplikacji mobilnej**

### **3.3.1 Android SDK**

Przy tworzeniu aplikacji mobilnej skorzystałem z Android SDK (software development kit), które miało swoją premierę 12 listopada 2007 roku. Aplikacja może być uruchomiona na urządzeniach z systemem Android od wersji 6.0 w górę. Android SDK jest zbiorem bibliotek i narzędzi potrzebnych do tworzenia i debugowania aplikacji.

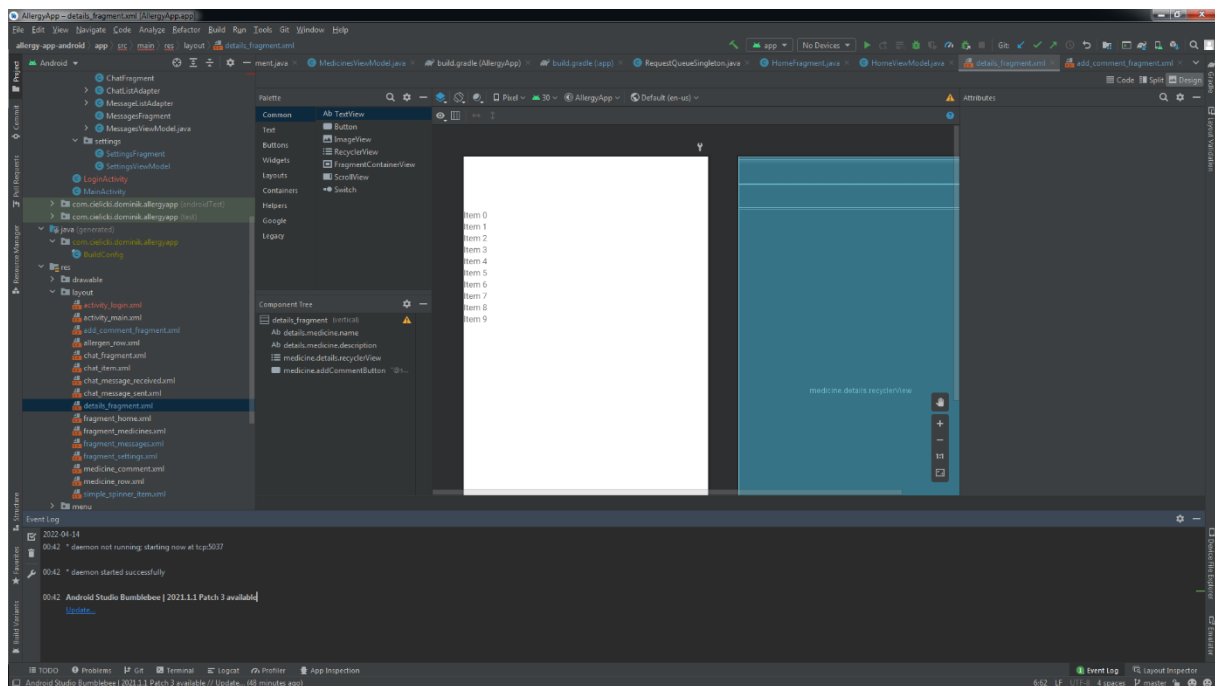
### **3.3.2 Android Studio**

Jako środowisko programistyczne wykorzystałem Android Studio<sup>6</sup>, które jest dedykowanym narzędziem do rozwoju aplikacji na system Android. Android Studio zawiera szereg udogodnień, które w znaczący sposób przyspieszają pracę. Należą do nich między innymi intuicyjny edytor graficzny, debugger oraz mechanizm podpowiadania składni.

---

<sup>5</sup> <https://www.jetbrains.com/idea/>

<sup>6</sup> <https://developer.android.com/studio>



Rysunek 16 Android Studio

### 3.3.3 Biblioteka Volley

Do wysyłania żądań HTTP zastosowałem bibliotekę Volley<sup>7</sup>. Została ona stworzona przez firmę Google w roku 2013. Funkcjonowanie biblioteki opiera się na zasadzie kolejki zapytań, która w miarę możliwości wysyła zapytania. W momencie tworzenia zapytania, definiujemy kod, który ma się wykonać w przypadku odpowiedzi z serwera.

### 3.3.4 Gradle

Gradle jest narzędziem do zarządzania i budowania projektów, podobnym do opisywanego wcześniej Maven'a, którego pierwsza wersja została wydana w 2008 roku przez HansaDocktera. Gradle'a oparty jest na plikach konfiguracyjnych build.gradle.

Gradle potrafi również pobierać biblioteki z repozytorium maven'a, co widać przy imporcie modułu allergy-app-commons.

<sup>7</sup> <https://google.github.io/volley/>

## 4. Implementacja aplikacji

### 4.1 Implementacja serwera REST

#### 4.1.1 Pliki konfiguracyjne

Spring framework wykorzystuje plik `application.properties` do przechowywania konfiguracji aplikacji.

```
spring.datasource.username=allergy-app
spring.datasource.password=allergy-app
spring.datasource.url=jdbc:hsqldb:file:allergyAppDB
spring.jpa.hibernate.ddl-auto=update
#spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
#security.ignored=/**
allergy_app.http.auth-token-header-name=X-ALLERGY-APP
allergy_app.http.auth-token=RNQdP1DabMnp8n0w
```

Rysunek 17 `application.properties`

W powyższej konfiguracji zauważyć możemy właściwości odpowiadające za ustawienia bazy danych, takie jak nazwa użytkownika, hasło, URL oraz tryb współpracy Hibernate z bazą. Dodatkowo w celu weryfikacji działania aplikacji ustawiony jest `spring.jpa.show-sql` na `true`, dzięki czemu w logach aplikacji widać wykonywane zapytania SQL.

Na samym końcu znajdują się ustawienia dotyczące zabezpieczenia serwera REST API. Tylko zapytania zawierające wskazane tokeny zostaną poprawnie przetworzone po stronie serwera. W przypadku braku tych tokenów zostanie zwrócona odpowiedź HTTP z kodem 401 (unauthorized).

#### 4.1.2 RestController

Do deklaracji punktów dostępowych serwera REST wykorzystujemy klasy z adnotacjami `@RestController`. Dodatkowo w adnotacjach `@RequestMapping` podajemy ścieżkę, na której ma nasłuchiwać definiowana przez nas usługa oraz typ przetwarzanych i zwracanych przez niego danych. Na poziomie metod wykorzystujemy adnotacje `@GetMapping` oraz `@PostMapping` wraz ze ścieżką, które określają jakiego

typu żądanie może zostać przetworzone przez poszczególne funkcje. Poszczególne parametry wejściowe funkcji mogą również zawierać adnotacje mówiące o miejscu, z jakiego mają być pobierane, w tym przypadku funkcji `addAllergen(@RequestBodyAllergen allergen)` oczekuje w ciele zapytania obiektu typu `Allergen` w formie jsona.

### 4.1.3 Klasy bazodanowe oraz compositekey

Wcześniej wspomniany Hibernate w celu komunikacji z bazą danych wymaga klas definiujących obiekty bazodanowe, tak zwane POJO (plain old Java object). Hibernate, podobnie jak Spring, korzysta z adnotacji. Adnotacja `@Entity` wskazuje, że klasa ma zostać potraktowana jako obiekt bazodanowy, natomiast `@Table` mówi nam, że jest to tabela. Mamy również możliwość dodawania adnotacji na poziomie pól, które decydują o właściwościach poszczególnych kolumn w tabeli.

Mamy również możliwość tworzenia bardziej skomplikowanych konstrukcji bazodanowych, takich jak tabele z kluczem złożonym. Adnotacja `@Embeddable` wskazuje, że klasa może zostać wykorzystana jako pole innego obiektu bazodanowego.

W docelowej klasie wskazujemy za pomocą adnotacji `@EmbeddedId` jako klucz główny utworzony wcześniej obiekt łączący tabele `medicine` oraz `user`.

### 4.1.4 Repository

W celu komunikacji z bazą danych możemy utworzyć interfejs, który rozszerza `JpaRepository`<sup>8</sup>. Mamy możliwość definiowania interesujących funkcji, dzięki Spring Framework nie musimy dostarczać implementacji dla nich, jest ona generowana automatycznie na podstawie nazwy funkcji. Poniższa funkcja wykona zapytanie:

```
SELECT * FROM chat WHERE user_id = userIdor user2_id = receiverId

public interface ChatRepository extends JpaRepository<Chat, Long> {
    List<Chat> findAllByUserIdOrUser2Id(Long userId, Long receiverId);
}
```

Sam interfejs `JpaRepository`, który rozszerzamy dostarcza już zdefiniowane funkcje:

---

<sup>8</sup> <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>

```

List<T> findAll();
List<T> findAll(Sort sort);
List<T> findAllById(Iterable<ID> ids);
<S extends T> List<S> saveAll(Iterable<S> entities);
void flush();
<S extends T> S saveAndFlush(S entity);
<S extends T> List<S> saveAllAndFlush(Iterable<S> entities);
default void deleteInBatch(Iterable<T>
entities){deleteAllInBatch(entities);}
void deleteAllInBatch(Iterable<T> entities);
void deleteAllByIdInBatch(Iterable<ID> ids);
void deleteAllInBatch();
T getOne(ID id);
T getById(ID id);
<S extends T> List<S> findAll(Example<S> example);
<S extends T> List<S> findAll(Example<S> example, Sort sort);

```

Mamy również możliwość definiowania własnych zapytań za pomocą adnotacji @Query.

```

public interface MedicineRepository extends JpaRepository<Medicine, Long> {
    @Query("SELECT avg(mc.rating) FROM MedicineComment mc WHERE
mc.id.medicine.id = ?1")
    BigDecimal getAverageScore(Long id);
}

```

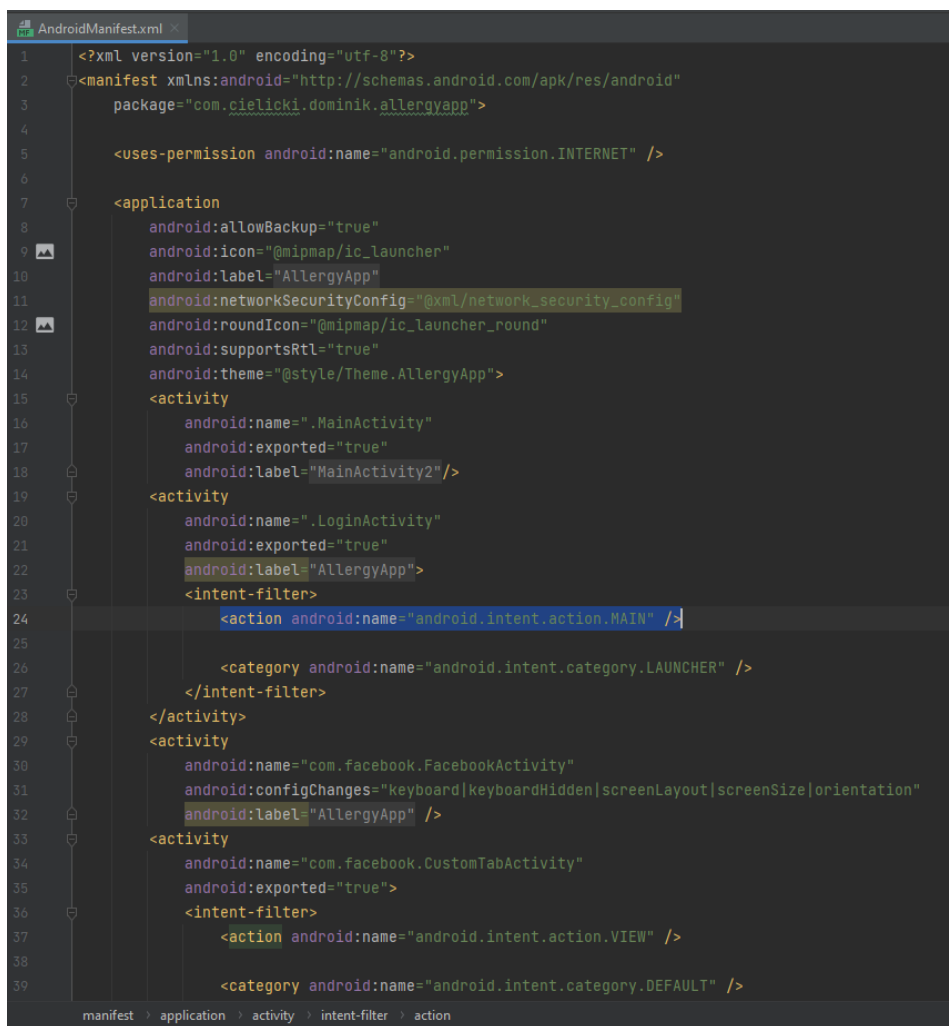
## 4.1.5 Service

Klasy z adnotacją @Service zawierają logikę biznesową, korzystają z interfejsów opisanych powyżej w celu komunikacji z bazą danych. Adnotacja @Autowired wskazuje, że instancja oczekiwanego obiektu, powinna zostać wstrzyknięta przez Spring'a.

## 4.2 Implementacja aplikacji mobilnej

### 4.2.1 Pliki konfiguracyjne

Konfiguracja aplikacji znajduje się w pliku `AndroidManifest.xml`. Definiujemy tam między innymi aktywności naszej aplikacji, ikonę, etykietę, ekran początkowy za pomocą tagu `<action android:name="android.intent.action.MAIN" />` oraz listę uprawnień potrzebnych do prawidłowego działania. Mamy również możliwość definiowania metadanych, które możemy wykorzystywać przy implementacji.



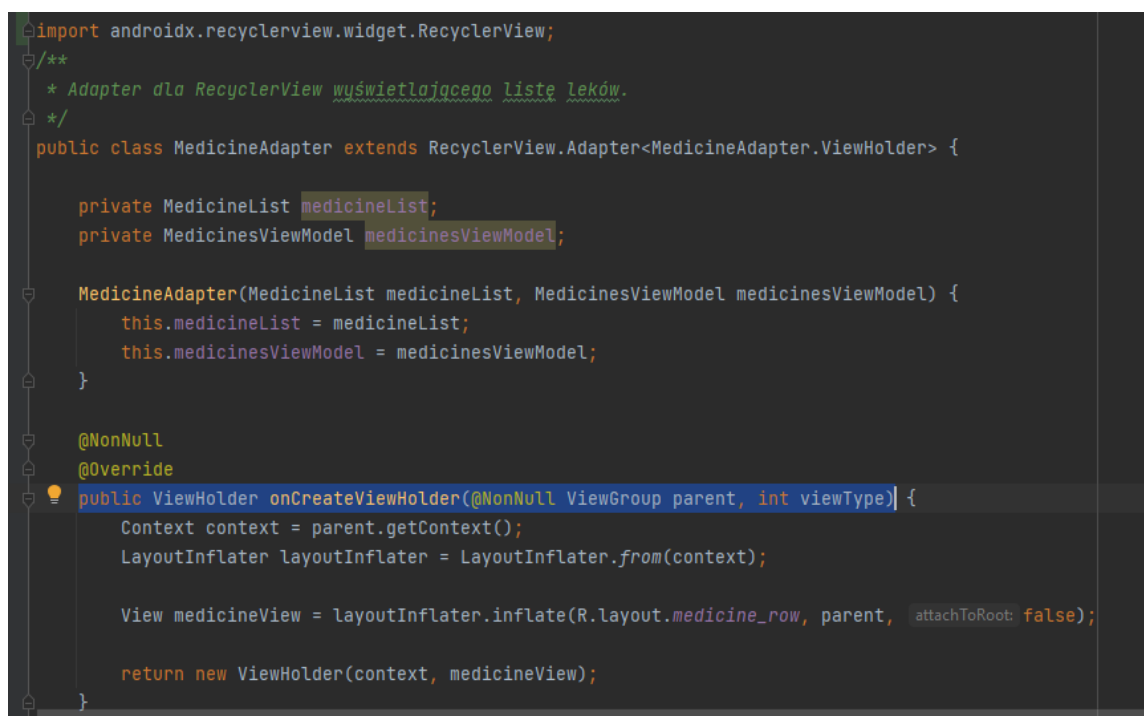
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.cielicki.dominik.allergyapp">
4
5     <uses-permission android:name="android.permission.INTERNET" />
6
7     <application
8         android:allowBackup="true"
9         android:icon="@mipmap/ic_launcher"
10        android:label="AllergyApp"
11        android:networkSecurityConfig="@xml/network_security_config"
12        android:roundIcon="@mipmap/ic_launcher_round"
13        android:supportsRtl="true"
14        android:theme="@style/Theme.AllergyApp">
15        <activity
16            android:name=".MainActivity"
17            android:exported="true"
18            android:label="MainActivity2"/>
19        <activity
20            android:name=".LoginActivity"
21            android:exported="true"
22            android:label="AllergyApp">
23            <intent-filter>
24                <action android:name="android.intent.action.MAIN" />
25                <category android:name="android.intent.category.LAUNCHER" />
26            </intent-filter>
27        </activity>
28        <activity
29            android:name="com.facebook.FacebookActivity"
30            android:configChanges="keyboard|keyboardHidden|screenLayout|screenSize|orientation"
31            android:label="AllergyApp" />
32        <activity
33            android:name="com.facebook.CustomTabActivity"
34            android:exported="true">
35            <intent-filter>
36                <action android:name="android.intent.action.VIEW" />
37                <category android:name="android.intent.category.DEFAULT" />
38            </intent-filter>
39        </activity>
40    </application>
41</manifest>
```

Rysunek 18 AndroidManifest.xml

## 4.2.2 RecyclerView

RecyclerView<sup>9</sup> jest komponentem do wyświetlania dynamicznych list. W celu wykorzystania tego komponentu musimy zaimplementować adapter, który definiuje zachowanie listy. Funkcja `onCreateViewHolder` wywoływana jest w momencie tworzenia elementu listy, natomiast funkcja `onBindViewHolder` wykonywana jest w celu uzupełniania danych lub definiowania zachowania w momencie wyświetlania.

Wygląd wiersza musi zostać zdefiniowany w postaci plików XML.

The image shows a code editor with Java code for a RecyclerView adapter. The code is as follows:

```
import androidx.recyclerview.widget.RecyclerView;

/**
 * Adapter dla RecyclerView wyświetlającego listę leków.
 */
public class MedicineAdapter extends RecyclerView.Adapter<MedicineAdapter.ViewHolder> {

    private MedicineList medicineList;
    private MedicinesViewModel medicinesViewModel;

    MedicineAdapter(MedicineList medicineList, MedicinesViewModel medicinesViewModel) {
        this.medicineList = medicineList;
        this.medicinesViewModel = medicinesViewModel;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        Context context = parent.getContext();
        LayoutInflater inflater = LayoutInflater.from(context);

        View medicineView = inflater.inflate(R.layout.medicine_row, parent, attachToRoot: false);

        return new ViewHolder(context, medicineView);
    }
}
```

Rysunek 19 RecyclerView

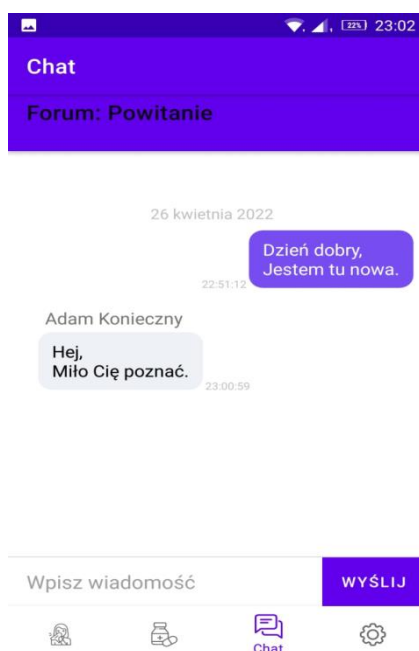
## 4.2.3 Implementacja Chatu

Chat również oparty jest na komponencie RecyclerView, posiada metodę `getItemViewType (int position)`, która decyduje o rodzaju wiadomości. W przypadku informacji przychodzącej wykorzystywany jest widok `chat_message_received.xml`, w przeciwnym wypadku wiadomość wyświetlana jest za pomocą widoku `chat_message_sent.xml`. W celu aktualizacji danych został stworzony wątek odpytujący co trzy sekundy serwer w poszukiwaniu aktualizacji. Wątek ten iteruje po liście chatów

<sup>9</sup> <https://developer.android.com/guide/topics/ui/layout/recyclerview#java>



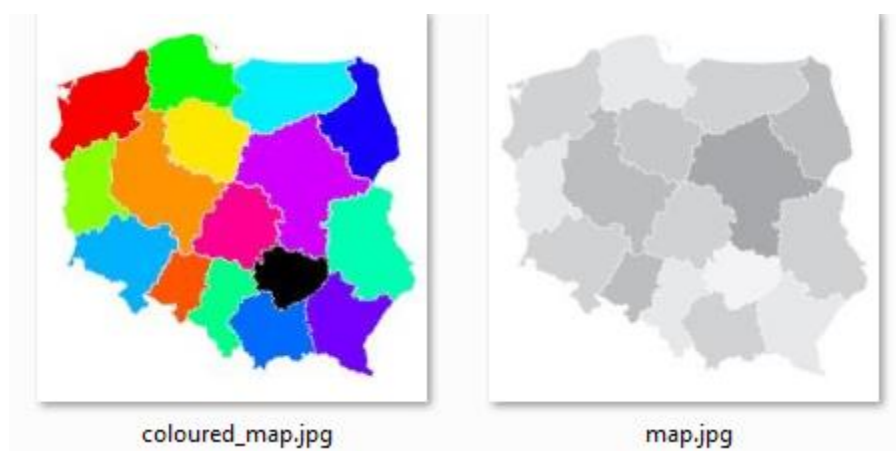
użytkownika i wysyła prośbę o listę wiadomości, które zostały przesłane o ID większym niż ostatnia otrzymana wiadomość.



Rysunek 20 Forum, "AllergyApp"

#### 4.2.4 Implementacja mapy

Mapa wyświetlana jest w komponencie `ImageView` jako jeden obrazek. Rozpoznawanie wskazanego województwa zostało zrealizowane za pomocą dwóch map. Pierwsza mapa, wyświetlana w aplikacji, jest w odcieniach szarości, natomiast druga jest pokolorowana unikalnym kolorem dla każdego z województw.



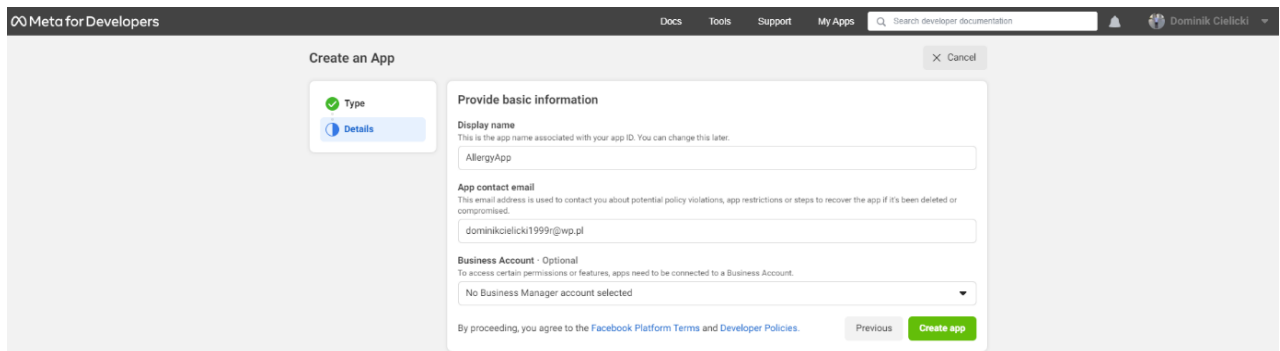
Rysunek 21 Widok mapy AllergyApp

Dodatkowo stworzyłem Enum, który opisuje poszczególne województwa. Dzięki funkcji `getVoivodeshipFromColor(int colorInt)` możemy pobrać obiekt województwa na podstawie koloru pobranego w miejscu kliknięcia na pierwszą mapę.

## 4.2.5 Implementacja logowania do Facebook

Na konfigurację logowania do FB składa się kilka elementów:

1. Rejestracja aplikacji na stronie Facebook'a<sup>10</sup>.



Rysunek 22 Rejestracja strony Facebook

2. Dodanie do pliku konfiguracyjnego Android Manifest aktywności oraz metadanych.
3. Dodanie kluczy do pliku z definicjami zmiennych tekstowych strings.xml:

```
<string name="facebook_app_id">713957716710523</string>
<string name="fb_login_protocol_scheme">fb1234</string>
<string name="facebook_client_token">193b768ff2a0d61deb977fc72371a3b2</string>
```

4. Dodanie zależności do pliku build.gradle:

```
implementation 'com.facebook.android:facebooklogin:latest.release'
```

5. Dodanie do ekranu logowania przycisku do logowania:

```
<com.facebook.login.widget.LoginButton
    android:id="@+id/login_button"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:gravity="center|center_horizontal|center_vertical"
    android:minHeight="44dp" />
```

W celu przechwytywania zdarzeń przycisku musimy zarejestrować FacebookCallback, który definiuje zachowanie w przypadku sukcesu, błędu lub anulowania logowania.

<sup>10</sup> <https://developers.facebook.com/>

```

CallbackManager callbackManager = CallbackManager.Factory.create();

LoginButton loginButton = (LoginButton)
findViewById(R.id.login_button);
loginButton.registerCallback(callbackManager, new
FacebookCallback<LoginResult>() {
    @Override
    public void onSuccess(LoginResult loginResult) {
        startLoading();
        getEmail(loginResult.getAccessToken());
    }

    @Override
    public void onCancel() {
        // Do nothing
    }

    @Override
    public void onError(FacebookException exception) {
        stopLoading();
        // Do nothing
    }
});

AccessToken accessToken = AccessToken.getCurrentAccessToken();
boolean isLoggedIn = accessToken != null &&
!accessToken.isExpired();

if (isLoggedIn) {
    startLoading();
    getEmail(accessToken);
}

```

Zdarzenie wylogowania się użytkownika przechwytyjemy za pomocą `AccessTokenTracker`, który reaguje na zmianę tokena wygenerowanego podczas logowania. Wylogowanie skutkować będzie restartem aplikacji.

## 4.2.6 ViewModel

Klasa `ViewModel` została stworzona w celu przetrzymywania i zarządzania danymi aplikacji uruchomionej na systemie Android. Wraz z wprowadzeniem klasy `ViewModel` dobrą praktyką stało się rozdzielanie warstwy widoku od warstwy danych. Instancje klas rozszerzających `ViewModel` mogą zostać pobrane za pomocą `ViewModelProvider`.

```

public class MessagesFragment extends Fragment {

    private MessagesViewModel messagesViewModel;

    private HomeViewModel mHomeViewModel;

    private FragmentMessagesBinding binding;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {

        messagesViewModel = new
ViewModelProvider(this).get(MessagesViewModel.class);

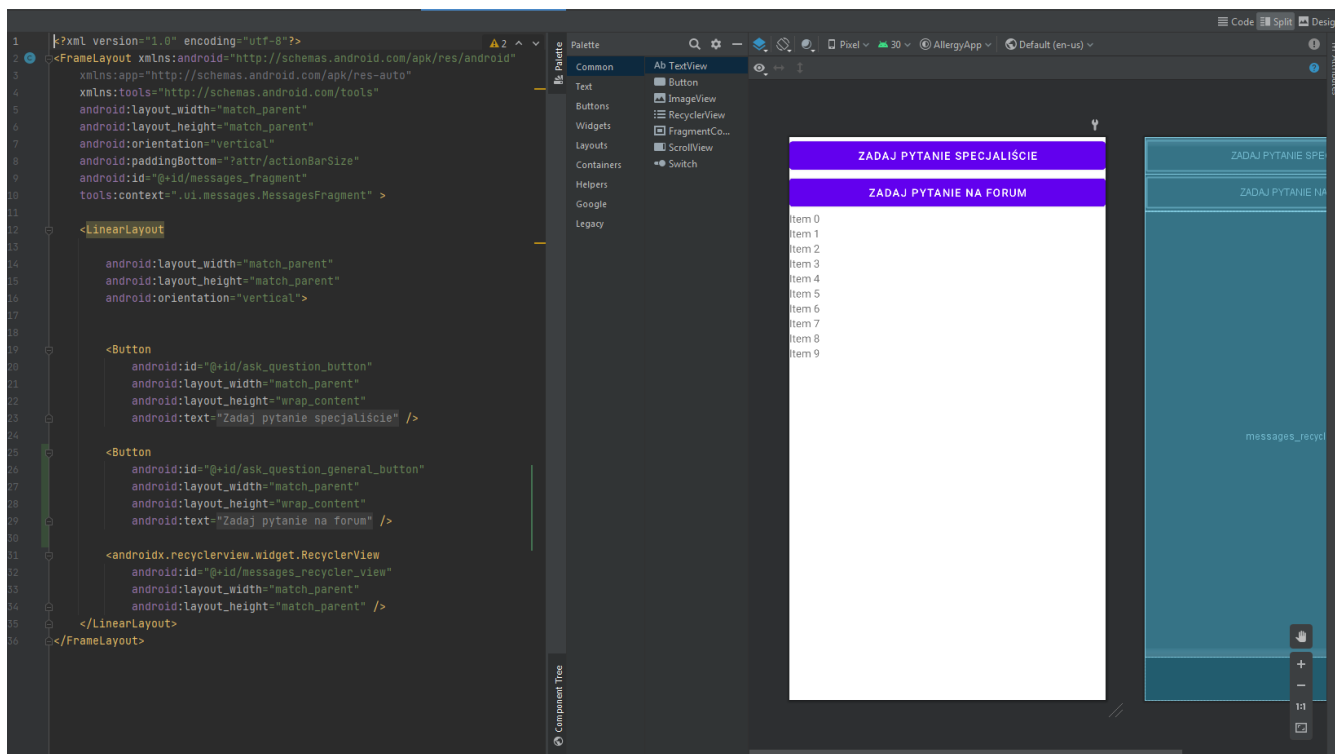
        mHomeViewModel = new
ViewModelProvider(getActivity()).get(HomeViewModel.class);

```

Jak widać na powyższym przykładzie konstruktor `ViewModelProvider` przyjmuje kontekst, w obrębie którego ma istnieć. Sprawdzane jest, czy w rzeczonym kontekście istnieje już instancja tej klasy. Jeśli tak, to jest zwracana, w przeciwnym wypadku obiekt jest inicjalizowany. Obiekt `messagesViewModel` stworzony jest w obrębie zakładki `MessagesFragment`, natomiast `HomeViewModel` tworzony jest na poziomie aktywności aplikacji.

## 4.2.7 Tworzenie wyglądu w XML

Interfejs użytkownika aplikacji definiowany jest w plikach XML. Android Studio wspiera graficzną edycję UI, pozwala na wybieranie elementów z zasobnika, zmianę właściwości, zmianę pozycji etc. Oprócz edycji graficznej, możliwa jest edycja bezpośrednio pliku XML.



Rysunek 23 Przykład widoku w XML

Komponenty mogą posiadać własność `android:id`, która wykorzystywana jest przy identyfikacji elementów z poziomu warstwy logicznej aplikacji.

## 4.2.8 Zmiana widoku w zakładce

W celu zmiany widoku z zakładki trzeba skorzystać z klasy `FragmentManager`. Pozwala on dodawać, usuwać oraz podmieniać interfejs użytkownika w ramach jednej zakładki.

```
Fragment fragment =
    ((AppCompatActivity)holder.context).getSupportFragmentManager().getFragment
    s().get(0);

FragmentManager fragmentManager = fragment.getChildFragmentManager();

fragmentManager.beginTransaction().replace(R.id.medicines_fragment,
    DetailsFragment.class,
    null).addToBackStack("details").setReorderingAllowed(true).commit();
```

Rysunek 24 Użycie klasy `FragmentManager`

Zgodnie z dokumentacją należy zawsze przy tworzeniu transakcji wywołać `setReorderingAllowed(true)`, spowodowane jest to wymogiem wstecznej kompatybilności. Metoda `commit()` jest asynchroniczna, co oznacza, że akcja zaplanowana w transakcji wykonana zostanie przy najbliższej przez główny wątek aplikacji. Dodatkowo w celu zapewnienia poprawnej nawigacji przy naciśnięciu przycisku cofania nadpisana została funkcja `onBackPressed()` w klasie `MainActivity`.

## 5. Aplikacja AllergyApp

### 5.1 Instalacja i uruchomienie serwera

Proces instalacji i uruchomienia serwera odbywa się na serwerze z publicznym adresem IP, dzięki takiemu podejściu korzystanie z aplikacji będzie możliwe dla wszystkich użytkowników Internetu. Pierwszym krokiem potrzebnym do uruchomienia serwera REST API jest zbudowanie pliku WAR za pomocą Maven'a:

```
mvn clean install
```

```
PS C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app (master)
> mvn clean install
[INFO] Scanning for projects...
[INFO]
[INFO] Reactor Build Order:
[INFO]
[INFO] Allergy App [pom]
[INFO] Allergy App Commons [jar]
[INFO] Allergy App REST Api [war]
[INFO]
[INFO] -----< com.cielicki.dominik:allergy-app >-----
[INFO] Building Allergy App 1.0.0 [1/3]
[INFO] -----[ pom ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ allergy-app ---
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ allergy-app ---
[INFO] Installing C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\pom.xml to C:\Users\domin\m2\repository\com\cielicki\dominik\allergy-app\1.0.0\allergy-app-1.0.0.pom
[INFO]
[INFO] -----< com.cielicki.dominik:allergy-app-commons >-----
[INFO] Building Allergy App Commons 1.0.0 [2/3]
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ allergy-app-commons ---
[INFO] Deleting C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-commons\target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ allergy-app-commons ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-commons\src\main\resources
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ allergy-app-commons ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 27 source files to C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-commons\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ allergy-app-commons ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-commons\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ allergy-app-commons ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ allergy-app-commons ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ allergy-app-commons ---
[INFO] Building jar: C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-commons\target\allergy-app-commons-1.0.0.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ allergy-app-commons ---
[INFO] Installing C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-commons\target\allergy-app-commons-1.0.0.jar to C:\Users\domin\m2\repository\com\cielicki\dominik\allergy-app-commons\1.0.0\allergy-app-commons-1.0.0.jar
[INFO] Installing C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-commons\pom.xml to C:\Users\domin\m2\repository\com\cielicki\dominik\allergy-app-commons\1.0.0\allergy-app-commons-1.0.0.pom
[INFO]
[INFO] -----< com.cielicki.dominik:allergy-app-rest-api >-----
[INFO] Building Allergy App REST Api 1.0.0 [3/3]
[INFO] -----[ war ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ allergy-app-rest-api ---
[INFO] Deleting C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-rest-api\target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ allergy-app-rest-api ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ allergy-app-rest-api ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 40 source files to C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-rest-api\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ allergy-app-rest-api ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ allergy-app-rest-api ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to C:\Users\domin\OneDrive\Pulpit\AllergyApp\allergy-app\allergy-app-rest-api\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ allergy-app-rest-api ---
```

Rysunek 25 Mvn clean install

W wyniku powyższej operacji powstał plik WAR, który trzeba uruchomić na serwerze aplikacyjnym Tomcat. W tym celu przenosimy plik zbudowany na komputerze na serwer docelowy. Jako serwer wybrałem maszynę z systemem operacyjnym Ubuntu<sup>11</sup>. Na serwerze Ubuntu zainstalowana została Java w wersji 8 oraz rozpakowany Apache Tomcat.

```
Last login: Tue Apr 26 22:11:14 2022 from 178.235.182.62
root@tomcat:~# cd /opt/apache-tomcat-9.0.62/
root@tomcat:/opt/apache-tomcat-9.0.62# ls
bin BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps work
root@tomcat:/opt/apache-tomcat-9.0.62# ls webapps/
allergy-app allergy-app.war docs examples host-manager manager ROOT
```

Rysunek 26 Serwer zewnętrzny

W katalogu `/opt/apache-tomcat-9.0.62/webapps/` umieszczamy zbudowany wcześniej plik WAR. Następnym krokiem jest uruchomienie serwera aplikacyjnego Tomcat za pomocą komendy `./catalina.sh run` lub `./catalina.sh start`. Wywołanie komendy z opcją `run` uruchamia serwer w bieżącym wątku konsoli, natomiast opcja `start` uruchamia serwer w tle.

```
Spring Boot (v2.6.0)
2022-04-26 22:19:10.197 INFO 19451 --- [main] c.c.d.a.ServletInitializer : Starting ServletInitializer using J
ava 11.0.14.1 with PID 19451 (/opt/apache-tomcat-9.0.62/webapps/allergy-app/WEB-INF/classes started by root in /opt/apache-tomcat-9.0.62
/bin)
2022-04-26 22:19:10.200 INFO 19451 --- [main] c.c.d.a.ServletInitializer : No active profile set, falling back
to default profiles: default
2022-04-26 22:19:11.283 INFO 19451 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repos
itories in DEFAULT mode.
2022-04-26 22:19:11.369 INFO 19451 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository sca
nning in 75 ms. Found 12 JPA repository interfaces.
2022-04-26 22:19:11.781 INFO 19451 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initial
ization completed in 1488 ms
2022-04-26 22:19:12.183 INFO 19451 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUn
itInfo [name: default]
2022-04-26 22:19:12.245 INFO 19451 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core versi
on 5.6.1.Final
2022-04-26 22:19:12.502 INFO 19451 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Anno
tations {5.1.2.Final}
2022-04-26 22:19:12.638 INFO 19451 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-04-26 22:19:13.096 INFO 19451 --- [main] hsqldb.db.HSQLDB8063D0009E.ENGINE : checkpointClose start
2022-04-26 22:19:13.099 INFO 19451 --- [main] hsqldb.db.HSQLDB8063D0009E.ENGINE : checkpointClose synched
2022-04-26 22:19:13.158 INFO 19451 --- [main] hsqldb.db.HSQLDB8063D0009E.ENGINE : checkpointClose script done
2022-04-26 22:19:13.160 INFO 19451 --- [main] hsqldb.db.HSQLDB8063D0009E.ENGINE : checkpointClose end
2022-04-26 22:19:13.166 INFO 19451 --- [main] com.zaxxer.hikari.pool.PoolBase : HikariPool-1 - Driver does not supp
ort get/set network timeout for connections. (feature not supported)
2022-04-26 22:19:13.172 INFO 19451 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-04-26 22:19:13.225 INFO 19451 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hiber
nate.dialect.HSQLDialect
2022-04-26 22:19:14.428 INFO 19451 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implem
entation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-04-26 22:19:14.431 INFO 19451 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactor
y for persistence unit 'default'
2022-04-26 22:19:14.561 WARN 19451 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled
by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable
this warning
```

Rysunek 27 Wywołanie `catalina.sh run`

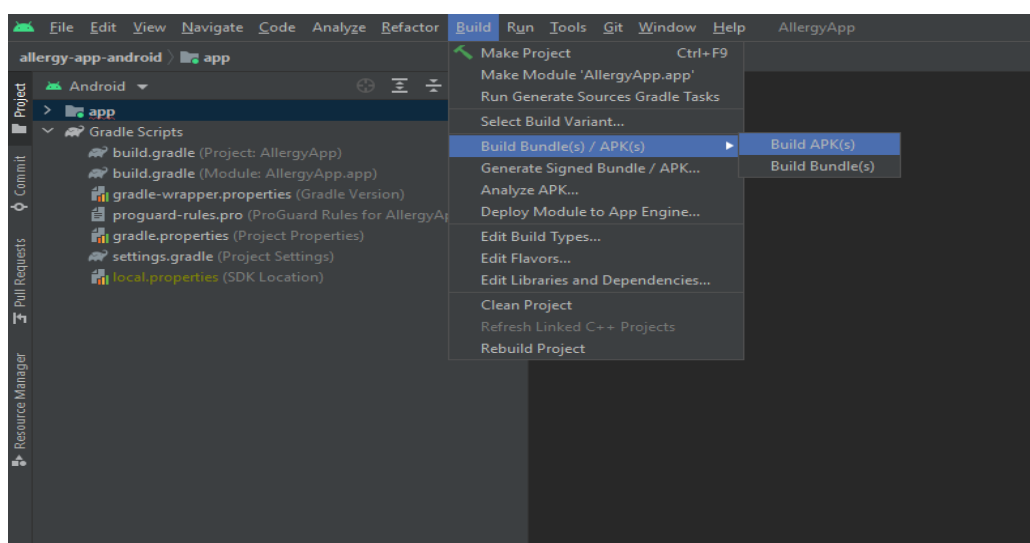
<sup>11</sup> <https://ubuntu.com/>



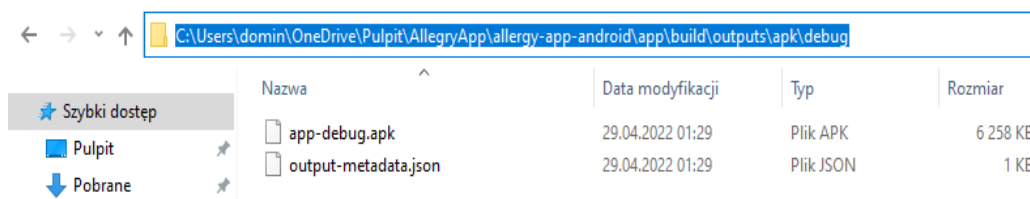
## 5.2 Uruchomienie aplikacji mobilnej

Aby uruchomić aplikację na telefonie, należy pobrać i zainstalować plik APK, który generowany jest z poziomu aplikacji Android Studio. IntelliJ IDEA umożliwia budowanie takich plików. W tym celu należy z menu wybrać opcję Build. Zostało to pokazane na zdjęciu poniżej. Po wykonaniu tej czynności, wymagany plik zostaje zbudowany i umieszczony w folderze debug, który znajdziemy przechodząc do pliku z aplikacją, a następnie:

`AllegryApp\allergy-app-android\app\build\outputs\apk\debug\app-debug.apk`



Rysunek 28 Build APK(s)

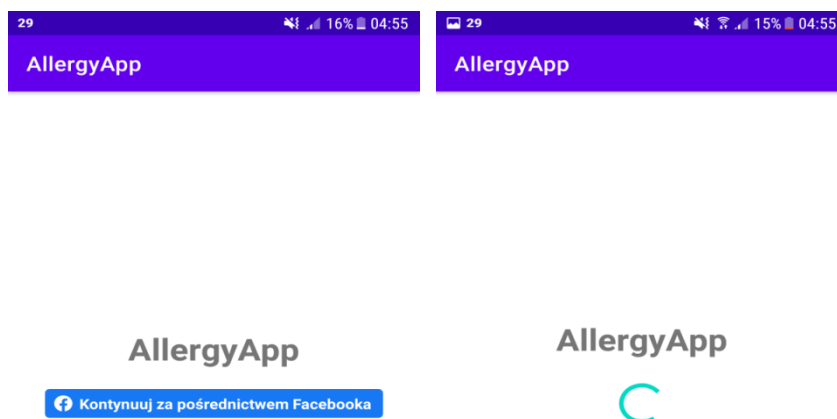


Rysunek 29 app-debug.apk

## 5.3 Wygląd i działanie zakładek aplikacji

### 5.3.1 Połączenie z kontem Facebook

Aplikacja AllergyApp po właściwym zainstalowaniu i uruchomieniu wyświetla interfejs do logowania. Jest to wymagany etap, aby móc korzystać z aplikacji. Wyświetlany jest ekran do logowania za pośrednictwem serwisu społecznościowego Facebook. Po poprawnym zalogowaniu następuje buforowanie, po czym jesteśmy przeniesieni do zakładki startowej, czyli Alergeny.



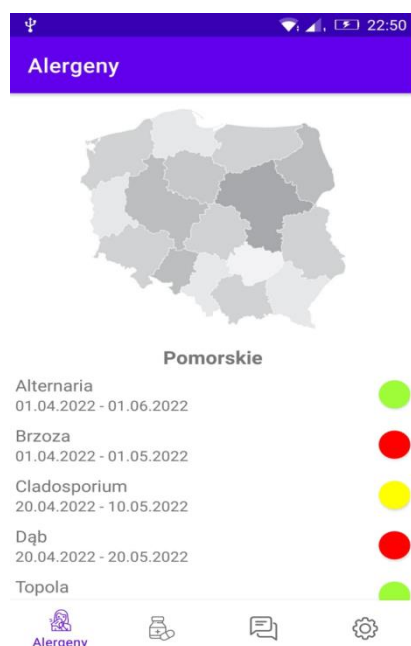
Rysunek 30 Logowanie, "AllergyApp"

Rysunek 31 Ładowanie, "AllergyApp"

### 5.3.2 Widok zakładki Alergeny

W zakładce alergenów wyświetlana jest mapa Polski, podzielona na województwa. Sposób działania mapy został opisany w rozdziale 4.2.4. Dotknięcie poszczególnych województw pobiera informacje o alergenach dla dzisiejszej daty. W celu optymalizacji działania aplikacji w klasie `HomeViewModel` dodana została `HashMap<Long, VoivodeshipAllergenList>`, która przechowuje już pobrane informacje. Przy każdym alergenie widoczny jest kolorowy wskaźnik. Aplikacja informuje o natężeniu alergenów w sposób obrazowy, korzystając z trzech kolorów. Kolor zielony, żółty oraz czerwony

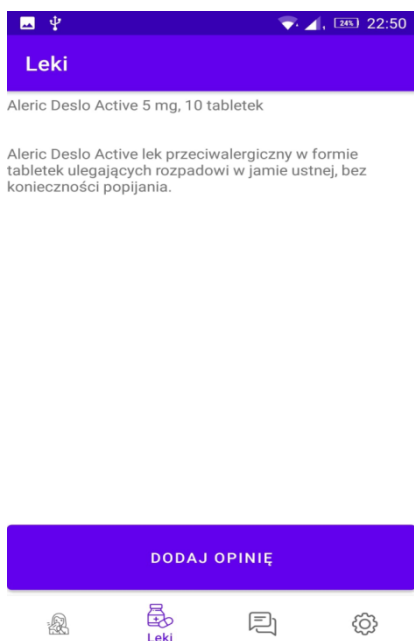
odpowiednio informują o natężeniu niskim, średnim oraz wysokim. Podany jest również zakres obowiązywania wskazanego pomiaru.



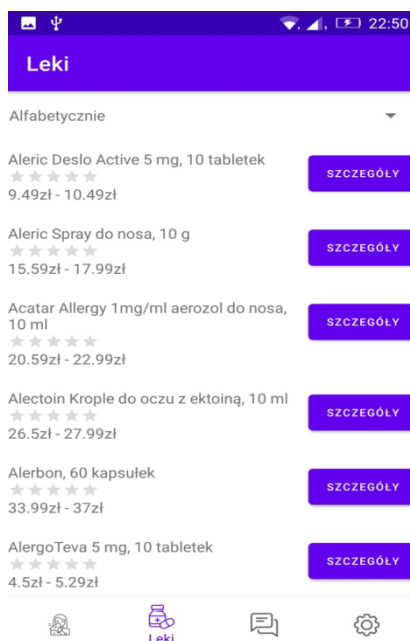
**Rysunek 32 Widok zakładki Alergeny,  
"AllergyApp"**

### 5.3.3 Widok zakładki Leki

Druga zakładka zawiera listę leków, wraz ze średnią ocen użytkowników. Przy każdym leku widoczna jest również informacja o przedziale cenowym danego produktu. Po wejściu w szczegóły leku widzimy opis oraz listę wystawionych komentarzy wraz z ocenami. Taki sposób przedstawienia leków pomaga użytkownikom w dokonaniu wyboru leku do walki z alergią. Warto przypomnieć jeszcze, że użytkownicy wystawiający oceny zweryfikowani są za pośrednictwem portalu społecznościowego Facebook.



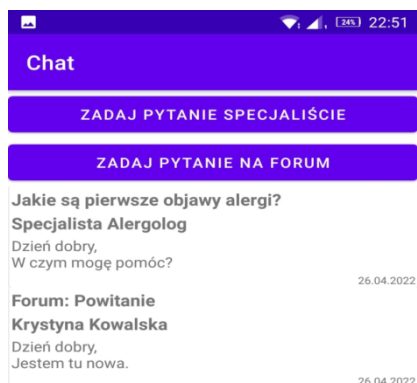
**Rysunek 33 Widok dodawania opinii, "AllergyApp"**



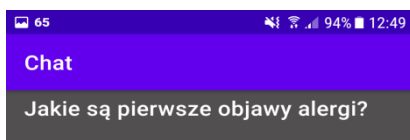
**Rysunek 34 Widok zakładki Leki, "AllergyApp"**

### 5.3.4 Widok zakładki Chat

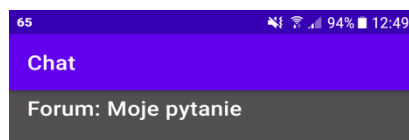
W zakładce z wiadomościami mamy możliwość zadania pytania specjalście lub założenie wątku na forum globalnym. Ponadto wyświetlana jest lista już otrzymanych wiadomości. Funkcja zadawania pytań skierowana jest do użytkowników, którzy potrzebują szczegółowych informacji na interesujący ich temat. Rozmowa przebiega tylko między specjalistą, a osobą pytającą. Jej przebieg nie jest wyświetlany dla innych użytkowników aplikacji. Dzięki takiej formie komunikacji jesteśmy w stanie skonsultować się w szybki sposób i uzyskać indywidualną poradę od kompetentnej osoby. Natomiast opcja zadawania pytań na forum daje możliwość udzielenia odpowiedzi lub utworzenie nowego wątku, skierowanego do każdego użytkownika aplikacji. Dzięki takiej funkcjonalności docieramy do szerokiej grupy odbiorców, co otwiera drogę na możliwość dzielenia się swoimi doświadczeniami w wygodny sposób.



Rysunek 35 Widok zakładki Chat, "AllergyApp"



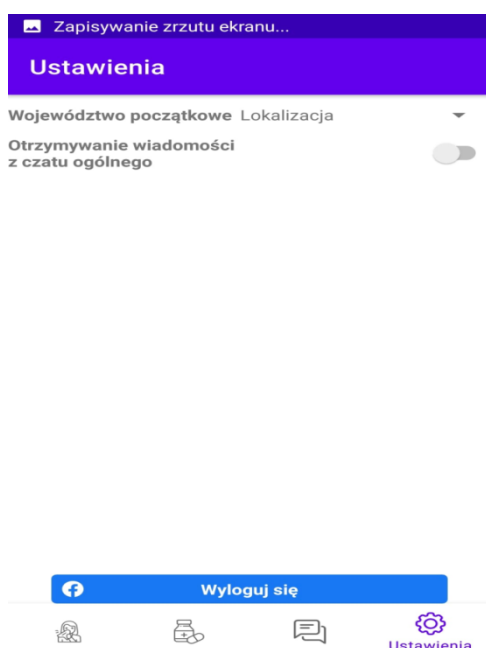
Rysunek 36 Tworzenie Chatu ze specjalistą, "AllergyApp"



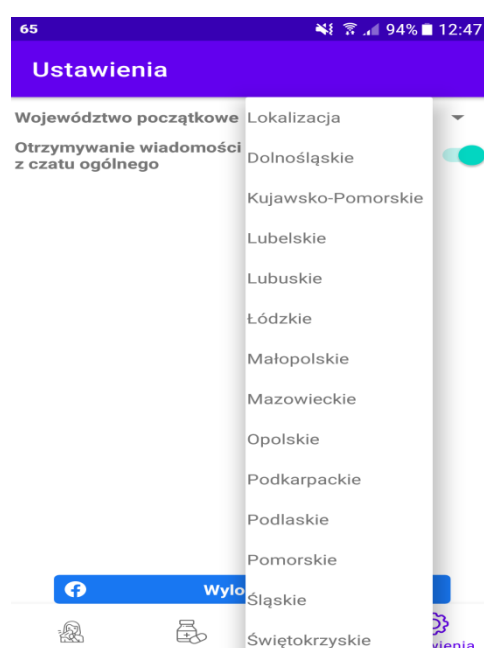
Rysunek 37 Tworzenie Chatu na forum, "AllergyApp"

### 5.3.5 Widok zakładki Ustawienia

Ostatnia zakładka zawiera ustawienia. W tej chwili możliwe jest wybranie lokalizacji początkowej, która ustawiana jest w momencie uruchomienia aplikacji. Do wyboru mamy wszystkie województwa oraz lokalizację GPS. Dodana została również możliwość decydowania o chęci otrzymywania wiadomości z ogólnego czatu. Dodany został również przycisk do wylogowania się z aplikacji. Po pomyślnym wylogowaniu aplikacja uruchamiana jest ponownie, dając nam możliwość ponownego zalogowania.



Rysunek 38 Widok zakładki Ustawienia, "AllergyApp"



Rysunek 39 Widok Województwo początkowe, "AllergyApp"

## 6. Podsumowanie

Celem pracy inżynierskiej było stworzenie aplikacji mobilnej na system Android przeznaczonej dla alergików, która korzysta z REST API. Miała ona na celu wspomóc użytkowników w wyborze leków, a także przekazać niezbędne informacje o stanie stężenia alergenów w województwach. W obecnej wersji aplikacja zasilona jest danymi statycznymi. W przypadku dalszej pracy nad aplikacją rozważyłbym pobieranie informacji o stężeniach alergenów ze stacji obserwacyjnych. Dodatkowo rozszerzeniu mogłaby ulec zakładka ustawień w celu lepszego dopasowania zakresu funkcjonalności, takich jak powiadomienia o wzroście stężeń wybranych alergenów w okolicy. W celu uaktywnienia komunikacji między użytkownikami warto by było również zaimplementować możliwość wysyłania emotikonów oraz plików multimedialnych takich jak gif lub zdjęcia. Zasadnym wydaje się również stworzenie zakładki umożliwiającej dzielenie się opiniami na forum o gabinetach alergologicznych, co ułatwiałaby wybór lekarza.

Uważam, że udało mi się osiągnąć zamierzony cel w ramach przyjętej konwencji. Jestem przekonany, że podobne aplikacje powinny być nieustannie modyfikowane, ponieważ jest jeszcze wiele niedopracowanych elementów w konkurencyjnych aplikacjach. W mojej ocenie projekt, którego jestem autorem, jest użyteczny i może mieć praktyczne zastosowanie w przypadku osób cierpiących na alergię. Aplikacja AllergyApp ma szansę zostać pełnowartościową aplikacją po uwzględnieniu modyfikacji powyżej wskazanych.

## 7. Bibliografia

- 1) <https://www.java.com/pl/>
- 2) <https://git-scm.com/>
- 3) <https://tomcat.apache.org/>
- 4) <https://maven.apache.org/>
- 5) <https://www.jetbrains.com/idea/>
- 6) <https://developer.android.com/studio>
- 7) <https://google.github.io/volley/>
- 8) <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>
- 9) <https://developer.android.com/guide/topics/ui/layout/recyclerview#java>
- 10) <https://developers.facebook.com/>
- 11) <https://ubuntu.com/>

## 8. Spis rysunków

Rysunek 1 Mapa pyleń, "Apsik" .....	6
Rysunek 2 Strona główna, "Apsik" .....	6
Rysunek 3 Mapa gabinetów, "Apsik" .....	6
Rysunek 4 Menu, "Momester Nasal" .....	7
Rysunek 5 Mapa pyleń, "Momester Nasal" .....	7
Rysunek 6 Dzienniczek objawów, "Momester Nasal" .....	7
Rysunek 7 Wykres stężenia pyłków, "MASK-air" .....	8
Rysunek 8 Panel główny, "MASK-air" .....	8
Rysunek 9 Architektura API REST .....	9
Rysunek 10 Przykład PostMapping .....	11
Rysunek 11 Java Virtual Machine (JVM) .....	12
Rysunek 12 Git log .....	13
Rysunek 13 Apache-tomcat, folder bin .....	14
Rysunek 14 pom.xml .....	15
Rysunek 15 Format JSON, Postman .....	17
Rysunek 16 Android Studio .....	19
Rysunek 17 application.properties .....	20
Rysunek 18 AndroidManifest.xml .....	23
Rysunek 19 RecyclerView .....	24
Rysunek 20 Forum, "AllergyApp" .....	25
Rysunek 21 Widok mapy AllergyApp .....	25
Rysunek 22 Rejestracja strony Facebook .....	26
Rysunek 23 Przykład widoku w XML .....	29
Rysunek 24 Użycie klasy FragmentManager .....	29
Rysunek 25 Mvn clean install .....	31
Rysunek 26 Serwer zewnętrzny .....	32
Rysunek 27 Wywołanie catalina.sh run .....	32
Rysunek 28 Build APK(s) .....	33
Rysunek 29 app-debug.apk .....	33
Rysunek 30 Logowanie, "AllergyApp" .....	34
Rysunek 31 Ładowanie, "AllergyApp" .....	34
Rysunek 32 Widok zakładki Alergeny, "AllergyApp" .....	35
Rysunek 33 Widok dodawania opinii, "AllergyApp" .....	36
Rysunek 34 Widok zakładki Leki, "AllergyApp" .....	36
Rysunek 35 Widok zakładki Chat, "AllergyApp" .....	37
Rysunek 36 Tworzenie Chatu ze specjalistą, "AllergyApp" .....	37
Rysunek 37 Tworzenie Chatu na forum, "AllergyApp" .....	37
Rysunek 38 Widok zakładki Ustawienia, "AllergyApp" .....	37
Rysunek 39 Widok Województwo początkowe, "AllergyApp" .....	37