

## Spis treści

Wstęp.....	1
- Obsługa Wielowątkowości .....	2
- Połączenie z serwerem oraz komunikacja .....	2
- Wysyłanie aktualnych obostrzeń.....	4
- Wysyłanie informacji o położeniu sanepidu i nr telefonu .....	6
- Wysyłanie objawów Covid-19 .....	9
- Wysyłanie informacji o aktualny statystykach .....	11
- Aktualne zarażenia .....	17

---

# KORONES

---

## Wstęp

Przedstawiamy dokumentację aplikacji „Korones”, program służący do informowania ludności w zakresie koronawirusa oraz postępowania w razie zakażenia.

Gdy tylko się dowiedzieliśmy o konkursie chcieliśmy wpasować się w aktualny trend. Uznaliśmy że zrobienie aplikacji która nie będzie nikomu przydatnie jest bez celu. Dlatego też postawiliśmy postawić na temat Koronawirusa. Do tego chcieliśmy zastosować ciekawą metodę tworzenia aplikacji- połączenie 2 języków programowania, w naszym przypadku Java i Python. Serwer został napisany w Pythonie a aplikacja w Javie. Uznaliśmy że do naszego projektu najlepsze będą serwery Gogola tzw. GCloud. Serwery te pozwalają na wielką swobodę oraz dowolność kodu.

Głównym naszym celem było połączenie oraz zebranie najważniejszych informacji w jednym miejscu. Naszą aplikację cechuje możliwość aktualizowania aktualnych wydarzeń bez udziału użytkownika. Kolejnym wielkim plusem jest to że aplikacja jest aktualizowana praktycznie co chwila, oznacza to że użytkownik jest cały czas informowany o nowych informacjach. Kod aplikacji jest w pełni skomentowany także jeżeli byłyby jeszcze jakieś niejasności można tam zajrzeć.

Wykorzystane aplikacje przy projekcie:

- Pycharm Community Edition
- Android studio
- Adobe Xd
- Gimp
- Eclipse

Autorzy:

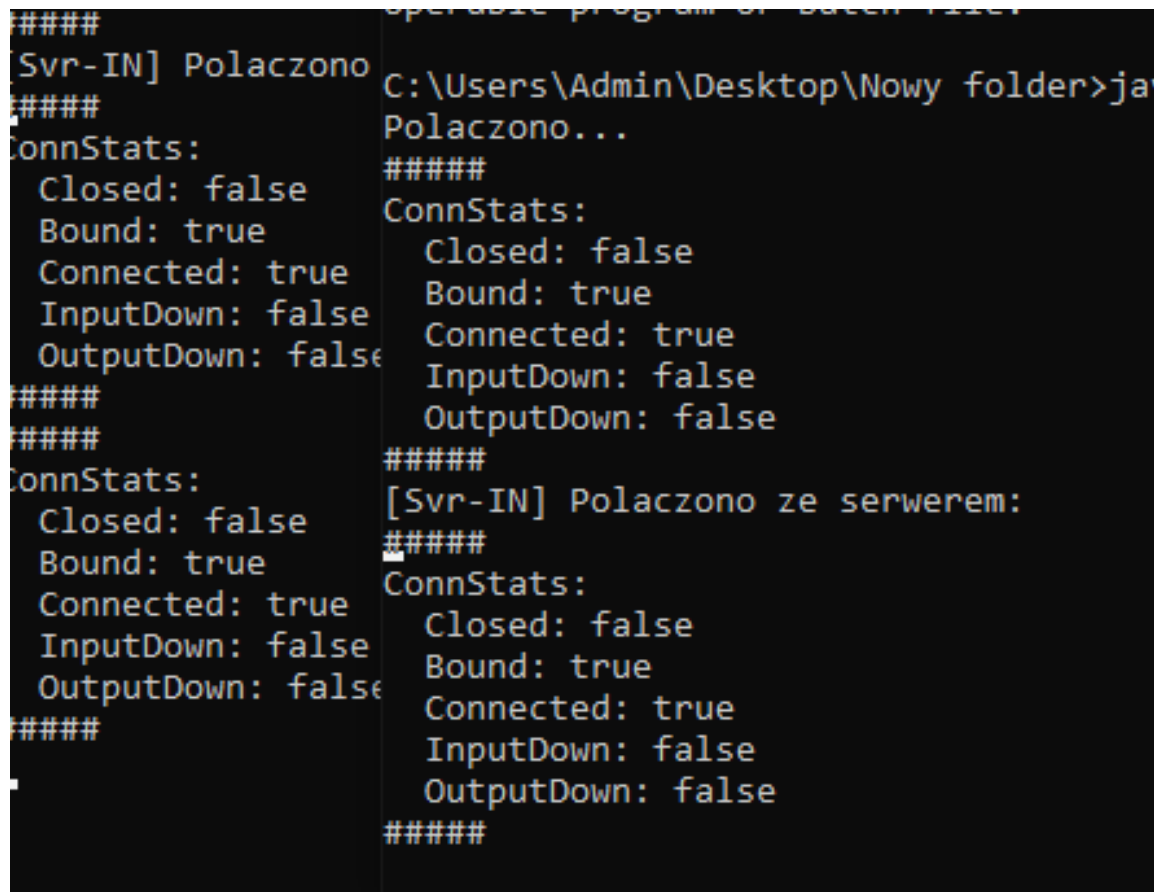
Patryk Migaj

Marcin Kryjom

Na potrzeby Serwera stworzyliśmy prosty program do jego testowania. Był on napisany w języku Java w postaci pliku: client.jar (Zrzuty ekranu z konsoli).

## - Obsługa Wielowątkowości

Oznacza to że jesteśmy przygotowani na wielu użytkowników naszej aplikacji. Każdy dostanie dokładnie takie same informacje zwrotne. Wykorzystaliśmy do tego celu moduły takie jak socket oraz threading.



```
#####
[Svr-IN] Polaczono
#####
ConnStats:
  Closed: false
  Bound: true
  Connected: true
  InputDown: false
  OutputDown: false
#####
#####
ConnStats:
  Closed: false
  Bound: true
  Connected: true
  InputDown: false
  OutputDown: false
#####
#####
[Svr-IN] Polaczono ze serwerem:
#####
ConnStats:
  Closed: false
  Bound: true
  Connected: true
  InputDown: false
  OutputDown: false
#####
#####
```

Kod Serwera:

```
class Klient(threading.Thread):
    def __init__(self, clientAddress, clientsocket):
        threading.Thread.__init__(self)
        self.csocket = clientsocket
        list.append(clientsocket)
        print("Nowe polaczenie od", clientAddress)
    def run(self):
        print("Polaczono z: ", clientAddress)
        self.csocket.send(bytes("Polaczono ze serwerem:" + '\n'))
        msg = ''
        #####
```

Autorzy:  
Patryk Migaj  
Marcin Kryjom

## - Połączenie z serwerem oraz komunikacja

Przy włączeniu aplikacji program próbuje się połączyć z serwerem poprzez stworzenie instancji stworzonej przez nas klasy o nazwie "DataSocketConnection.java". Odpowiada ona za łączenie i utrzymywanie połączenia, służy do wysyłania i odbierania informacji.

```
[Korones-LOG] cinfo :=> Tworzenie połączenia
[Korones-LOG] cstate :=> OPENING
[Korones-LOG] cstate :=> OPEN
[Korones-LOG] cinfo :=> Pętla ping
[Korones-LOG] cmdsend :=> /ping;1591638615719
[Korones-LOG] recive :=> Polaczono ze serwerem:
[Korones-LOG] cmdsend :=> /zarazenia
[Korones-LOG] recive :=> /pong;1591638615719
[Korones-LOG] pingtime :=> 53
```

W konsoli wyświetlane są powyższe informacje.

Następnie tworzona jest pętla ping która informuje aplikację o połączeniu z serwerem.

Serwer odczytuje to i odsyła komendę /pong;czas

Jeżeli chodzi o serwer to skonfigurowany jest tak aby pracował na linuxie. W naszym wypadku jest to debian.

Kod Serwera:

```
if '/ping;' in msg:
    ping = msg.split(';')
    pong = '/pong;' + ping[1] + '\n'
    self.csocket.send(pong.encode('UTF-8'))
```

Nasza aplikacja bazuje tak naprawdę na zasadzie chatu z serwerem. Możemy sobie przysyłać pozdrowienia itp. Praktycznie jednak nasz klient robi wszystko automatycznie i nic wpisywać nie trzeba. Funkcja wywołuje się w momencie uruchomienia aplikacji i uruchamia nowy wątek.

Praktyczne działanie:

```
[OUT] Witam serdecznie to nasz tekst na domuneetacje
[Svr-IN] witam rowniez
####
ConnStats:
  Closed: false
  Bound: true
  Connected: true
  InputDown: false
  OutputDown: false
####
Czekanie na klienta
('Nowe polaczenie od', ('185.22.8.72', 62692))
('Polaczono z: ', ('185.22.8.72', 62692))
('from client', '', ('185.22.8.72', 62692), '', u'Witam serdecznie to nasz tekst na domuneetacje\r\n')
witam rowniez
```

Autorzy:  
Patryk Migaj  
Marcin Kryjom

## - Wysyłanie aktualnych obostrzeń

Gdy wciśniemy odpowiedni przycisk to pierwsze co się dzieje to włącza się odpowiedni layout oraz program wysyła komendę /obostrzenia.

```
DataManager.getDataSocketConnection().putCallback( key: "stateupdate", new Callback() {  
    @Override  
    public void dataInputBack(String key, String value) {  
        if(value.equals("OPEN")){  
            DataManager.getDataSocketConnection().send( cmd: "/obostrzenia");  
            DataManager.getDataSocketConnection().removeCallback(key, callback: this);  
        }  
    }  
});
```

Gdy przyjdzie informacja od aplikacji serwer uruchamia odpowiednią funkcję. Aby uzyskać informacje o obostrzeniach wybraliśmy stronę <https://www.gov.pl/web/koronawirus/3etap>

Kod:

```
# -*- coding: utf-8 -*-  
from bs4 import BeautifulSoup  
import requests  
import csv  
import sys  
import datetime  
import time  
def funkcja():  
    lista_kraje= []  
    source = requests.get('https://www.gov.pl/web/koronawirus/3etap').text  
    soup = BeautifulSoup(source, 'lxml')
```

Jak widać został wykorzystany moduł BeautifulSoup pozwalający na wycinanie pewnych informacji ze strony. Została zabrana klasa (w tym wypadku 'editor-content') i wysłana jako HTML

```
####  
obostrzenia  
OUT] /obostrzenia  
Svr-IN] <div class="editor-content">  
Svr-IN] /obo;<div><p><span style="font-size:11pt"><span style="font-size:12.0pt">Pierwsze poluzowanie obostrzeń wpr  
anych z powodu pandemii nastąpiło 20 kwietnia. Od tego czasu uważnie analizujemy, w jaki sposób znoszenie poszczegół  
ograniczeń wpływa na przyrost zachorowań i wydolność służby zdrowia. Co się zmieni w najbliższym czasie wraz z wej
```

Autorzy:

Patryk Migaj


Marcin Kryjom

W tym momencie gdy program otrzyma tekst z pierwszym tekstem /obo; automatycznie przekonwertuje go z HTML na tekst z zachowaniem wszelkich nagłówków, hiperłączy itp. (W zależności od wersji systemu inicjuje się inna funkcja)

Kod Aplikacji:

```
@SuppressWarnings("HandlerLeak")
final Handler handler = new Handler(){
    @Override
    public void handleMessage(Message msg) {
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N){
            num.append(Html.fromHtml(msg.getData().getString( key: "text"), flags: 0));
        }else{
            num.append(Html.fromHtml(msg.getData().getString( key: "text")));
        }
    }
};
```

Wygląd ostateczny:

 **Obostrzenia**

Pierwsze poluzowanie obostrzeń wprowadzanych z powodu pandemii nastąpiło 20 kwietnia. Od tego czasu uważnie analizujemy, w jaki sposób znoszenie poszczególnych ograniczeń wpływa na przyrost zachorowań i wydolność służby zdrowia. Co się zmieni w najbliższym czasie wraz z wejściem w trzeci etap znoszenia ograniczeń?

**Działalność gospodarcza – otwieramy salony kosmetyczne i fryzjerskie oraz restauracje**

Odmrażamy gastronomię i salony kosmetyczne. Od 18 maja będziemy mogli:

- skorzystać z usług kosmetyczek i fryzjerów,
- zjeść w restauracji, kawiarni, barze, a także w centrum handlowym (w wydzielonej przestrzeni gastronomicznej).

**Ważne!** Gość będzie mógł zjeść posiłek zarówno na sali, w środku, jak i w ogródku restauracyjnym.

Pełne otwarcie branży gastronomicznej musi odbyć się pod warunkiem przestrzegania ścisłych wytycznych sanitarnych, które zapewnią bezpieczeństwo i obsłudze, i klientom. Zasady te zostały określone podczas konsultacji z przedstawicielami poszczególnych branż.

**Wśród zaleceń dla salonów urody znalazły się m.in.:**

- obowiązek noszenia przez klientów i obsługę maseczek, gogli lub przyłbic (u fryzjera, a w salonie kosmetycznym – jeśli zabieg na to pozwala);
- używanie ręczników jednorazowych. ndr jest to

Autorzy:

Patryk Migaj

Marcin Kryjom

## - Wysyłanie informacji o położeniu sanepidu i nr telefonu

Kiedy klikniemy w przycisk "Kontakt z Sanepidem" wyświetli się odpowiedni layout. Powstanie tekst, oraz belka do wpisywania miejscowości. Do tej belki gdy wpisujemy miejscowość to program wysyła komend /numer;miejscowosc do serwera.

Konsola aplikacji:

```
[Korones-LOG] cmdsend :=> /numer;kalisz
```

Gdy serwer otrzyma informacje o miejscowości aktywuje funkcję. Funkcja działa na zasadzie automatycznym wpisywaniu frazy Sanepid psse kontakt [nazwa miejscowości] sanepid w google. Serwer poprzez bs4 wypisuje informacje o stacji oraz je przesyła.

Funkcja Serwera:

Następnie serwer wysła informacje które otrzymał z funkcji.

Kod Serwera:

```
if '/numer;' in msg:
    dachowa = msg.split(';')
    bariera = dachowa[1]
    satelita = NumerMazowsze.funkcja(bariera)
    satelita = str(satelita)
    self.csocket.send(satelita.encode('UTF-8'))
```

Kod Funkcji:

```
from bs4 import BeautifulSoup
import requests
import csv
import sys
import datetime
import time
import urllib

def funkcja(zmienna):
    source = requests.get('https://www.google.pl/search?client=firefox-b-d&q=Sanepid+psse+kontakt+telefon' + zmienna + 'sanepid' + '&lr=lang_pl').text
    soup = BeautifulSoup(source, 'lxml')

    try:
        match = soup.find_all('span', {'class': 'BNeawe tAd8D AP7Wnd'})
        #match = match[0].text
        match = match[len(match) - 1]
        match = match.text
        try:
            x = match
            x = x.replace(' ', '')
            x = int(x)
            match = str(match)
            xw = soup.find_all('div', {'class': 'BNeawe tAd8D AP7Wnd'})
            xw = xw[0]
            xw = xw.text
            xw = xw.split('\n')
            xw = xw[1]
            wh = soup.find_all('span', {'class': 'BNeawe tAd8D AP7Wnd'})
            wh = wh[0]
            wh = wh.text
            wx = '/nazwa;' + xw
            wx += '/ulica;' + wh
            wx += '/numer;' + match + '\n'
            return wx
        except:
            match = '/nazwa:null' + '\n'
            return match
    except:
        match = '/nazwa:null\n'
        return match
```

Autorzy:  
Patrik Migaj  
Marcin Kryjom

Output:

```
[Svr-IN] Polaczono ze serwerem:  
/number;kalisz  
[OUT] /number;kalisz  
[Svr-IN] /nazwa;Inspekcja sanitarna w Kaliszu, Polska/ulica;62-800 Kalisz, Poland/number;+48 62 767 76 10
```

Jeżeli google nic nie znajdzie, serwer wysyła null.

Kiedy nazwa miejscowości jest prawidłowa aplikacja otrzymuje wiadomość takiego typu:

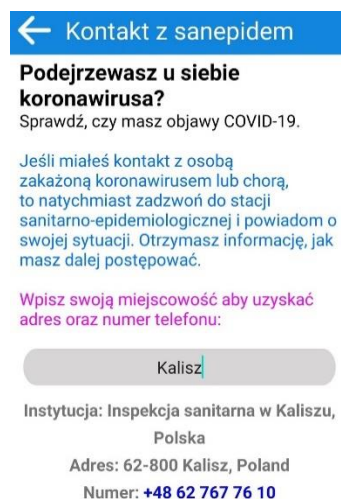
```
[Korones-LOG] recive :=> /nazwa;Inspekcja sanitarna w Kaliszu, Polska/ulica;62-800 Kalisz, Poland/number;+48 62 767 76 10
```

Następnie zamienia program wkłada to do odpowiedniego miejsca.

```
try{  
    String instytucja = value.split( regex: "/ulica")[0];  
    String adres = value.split( regex: "/ulica")[1].split( regex: "/number")[0].split( regex: "%")[1];  
    String number = value.split( regex: "/number")[1].split( regex: "%")[1];  
    Message message = notHandler.obtainMessage();  
    Bundle bundle = new Bundle();  
    bundle.putString("found", "Instytucja: "+instytucja+  
        "\nAdres: "+adres+  
        "\nNumer: ");  
    bundle.putString("number", number);  
    message.setData(bundle);  
    notHandler.sendMessage(message);  
    DatabaseHelper.setValue( context: ContactActivity.this, key: "region_cached_"+name, value);  
}catch (Exception ex) {  
    ex.printStackTrace();  
    Utils.Log("error", "Błąd podczas odbierania informacji (splitter)");  
}
```

Gdy aplikacja dostanie nieprawidłowe informacje aplikacja nie wyłączy się lecz zostanie wywołany wyjątek.

Wygląd końcowy:



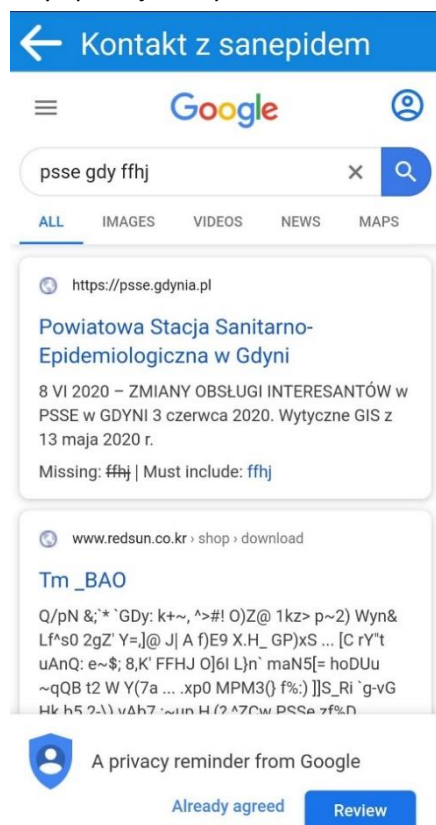
Tworzony jest przycisk pozwalający szybkie zadzwonienie do Sanepidu.

Autorzy:

Patryk Migaj

Marcin Kryjom

Gdy aplikacja otrzyma null otwierana jest przeglądarka z wyszukiwaniem wpisanej wcześniej miejscowości.





Autorzy:  
Patryk Migaj  
Marcin Kryjom

## - Wysyłanie objawów Covid-19

W tym miejscu po wejściu w objawy aplikacja wysyła do serwera komendę /objawy.

Kod Aplikacji:

```
DataManager.getDataSocketConnection().putCallback( key: "stateupdate", new Callback() {  
    @Override  
    public void dataInputBack(String key, String value) {  
        if(value.equals("OPEN")){  
            DataManager.getDataSocketConnection().send( cmd: "/objawy");  
            DataManager.getDataSocketConnection().removeCallback(key, callback: this);  
        }  
    }  
});
```

Następnie serwer wysyła informację zwrotną. Kod:

```
# -*- coding: utf-8 -*-  
import sys  
reload(sys)  
sys.setdefaultencoding('utf-8')  
def funkcja():  
    a = ("  
/objawy;<h4>COVID-19 wpływa na każdego w inny sposób. U większości zarażonych osób  
rozwinie się choroba o łagodnym lub umiarkowanym nasileniu. Takie osoby wyzdrowieją bez  
konieczności hospitalizacji.</h2>  
/objawy;\t  
/objawy;<h6>Najczęściej występujące objawy:</h6>  
/objawy;\t  
itd..  
return a
```

Po odebraniu następuje przekierowywanie do wątku głównego aby zmienić tekst oraz formatuje go z html na tekst.

Kod.

```
final Handler handler = new Handler(){  
    @Override  
    public void handleMessage(Message msg) {  
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N){  
            num.append(Html.fromHtml(msg.getData().getString( key: "text"), flags: 0));  
        }else{  
            num.append(Html.fromHtml(msg.getData().getString( key: "text")));  
        }  
    }  
};  
DataManager.getDataSocketConnection().putCallback( key: "/objawy", new Callback() {  
    @Override  
    public void dataInputBack(String key, String value) {  
        Message msg = handler.obtainMessage();  
        Bundle bundle = new Bundle();  
        bundle.putString("text",value);  
        msg.setData(bundle);  
        handler.sendMessage(msg);  
    }  
});
```

Autorzy:  
Patryk Migaj  
Marcin Kryjom  
Ostateczny wygląd

## ← Objawy

COVID-19 wpływa na każdego w inny sposób. U większości zarażonych osób rozwinię się choroba o łagodnym lub umiarkowanym nasileniu. Takie osoby wyzdrowieją bez konieczności hospitalizacji.

Najczęściej występujące objawy:

- gorączka
- suchy kaszel
- zmęczenie

Rzadziej występujące objawy:

- ból mięśni
- ból gardła
- biegunka
- zapalenie spojówek
- ból głowy
- utrata smaku lub węchu
- wysypka skórna lub przebarwienia palców u rąk i stóp

Poważne objawy:

- trudności w oddychaniu lub duszności
  - ból lub ucisk w klatce piersiowej
  - utrata mowy lub zdolności ruchowych
-

Autorzy:  
Patrik Migaj  
Marcin Kryjom

## - Wysyłanie informacji o aktualny statystykach

Kiedy klikniemy na przycisk wczytuję się layout. Mamy możliwość wpisania nazwy kraju o którym chcemy zobaczyć statystyki. Aplikacja sprawdza czy w lokalnej bazie danych nie ma zapisanej już listy krajów. Kiedy nie ma wysyła komendę /kraj do serwera.

```
DataManager.getDataSocketConnection().putCallback( key: "/kraj", new Callback() {  
    @Override  
    public void dataInputBack(String key, String value) {  
        DatabaseHelper.insertCountry(context,value);  
    }  
});  
DataManager.getDataSocketConnection().putCallback( key: "stateupdate", new Callback() {  
    @Override  
    public void dataInputBack(String key, String value) {  
        if(value.equals("OPEN")){  
            DataManager.getDataSocketConnection().send( cmd: "/kraj");  
            DataManager.getDataSocketConnection().removeCallback(key, callback: this);  
        }  
    }  
});
```

Kiedy serwer otrzyma komendę, wykonuje odpowiednia funkcję oraz wysyła nazwy wszystkich krajów w formie /kraj;Angielskanazwa;Polskanazwa

```
/kraj  
[OUT] /kraj  
[Svr-IN] /kraj;USA  
[Svr-IN] /kraj;Brazil;Brazylia  
[Svr-IN] /kraj;Russia;Rosja  
[Svr-IN] /kraj;Spain;Hiszpania  
[Svr-IN] /kraj;UK;Wielka Brytania  
[Svr-IN] /kraj;India;Indie  
[Svr-IN] /kraj;Italy;Włochy  
[Svr-IN] /kraj;Peru;Peru  
[Svr-IN] /kraj;Germany;Niemcy  
[Svr-IN] /kraj;Iran;Iran
```

Kod serwera:

```
1  # -*- coding: utf-8 -*-  
2  def funkcja():  
3      pow = ''  
17     return pow  
18
```

Kiedy aplikacja otrzyma informacje o krajach tworzy listę.

Następnie użytkownik jest proszony o wpisanie nazwy kraju. Funkcja sprawdza lokalna bazę danych czy już nie ma w niej zapisanego danego kraju w przeciągu 5 min

Jeżeli jest wypisuje dane z bazy danych

Autorzy:  
Patrik Migaj  
Marcin Kryjom  
Kod:

```
private void loadFromDatabase(String country){
    updateTextView(R.id.stats_text_allcases_val,DatabaseHelper.getValue( context: this, key: country+";allcases"));
    updateTextView(R.id.stats_text_newcases_val,DatabaseHelper.getValue( context: this, key: country+";newcases"));
    updateTextView(R.id.stats_text_alldeads_val,DatabaseHelper.getValue( context: this, key: country+";alldeads"));
    updateTextView(R.id.stats_text_newdeath_val,DatabaseHelper.getValue( context: this, key: country+";newdeath"));
    updateTextView(R.id.stats_text_allrecived_val,DatabaseHelper.getValue( context: this, key: country+";allrecived"));
    updateTextView(R.id.stats_text_activecases_val,DatabaseHelper.getValue( context: this, key: country+";activecases"));
    updateTextView(R.id.stats_text_critical_val,DatabaseHelper.getValue( context: this, key: country+";critical"));
    updateTextView(R.id.stats_text_cases1m_val,DatabaseHelper.getValue( context: this, key: country+";cases/1m"));
    updateTextView(R.id.stats_text_death1m_val,DatabaseHelper.getValue( context: this, key: country+";death/1m"));
    updateTextView(R.id.stats_text_alltest_val,DatabaseHelper.getValue( context: this, key: country+";alltest"));
    updateTextView(R.id.stats_text_test1m_val,DatabaseHelper.getValue( context: this, key: country+";test1/m"));
    updateTextView(R.id.stats_text_population_val,DatabaseHelper.getValue( context: this, key: country+";population"));
    updateTextView(R.id.stats_text_continent_val,DatabaseHelper.getValue( context: this, key: country+";continent"));
}
```

Kiedy nie ma w bazie to program wysyła komendę /info;kraj

```
DataManager.getDataSocketConnection().putCallback( key: "stateupdate", new Callback() {
    @Override
    public void dataInputBack(String key, String value) {
        if (value.equals("OPEN")) {
            DataManager.getDataSocketConnection().send( cmd: "/info;" + selected);
            DataManager.getDataSocketConnection().removeCallback(key, callback: this);
        }
    }
});
```

Kiedy serwer otrzyma odpowiednia komendę uruchamia funkcje:

Wykorzystaliśmy do tego zadania moduł o nazwie BeautifulSoup4. Pozwala on na uzyskanie informacji ze strony. Wybraliśmy stronę <https://www.worldometers.info/coronavirus/>. Przy użyciu tego modułu i formatowaniu mamy już pełną komendę do wysłania. Kod serwera:

```
# -*- coding: utf-8 -*-
from bs4 import BeautifulSoup
import requests
import csv
import sys
import datetime
import time
def funkcja():
    dzialy = ['North America', 'Europe', 'South America', 'Asia', 'Africa', 'Oceania', '', '\n', 'World','Total:']
    source = requests.get("https://www.worldometers.info/coronavirus/#c-all%22").text
    list = ['country', 'allcases', 'newcases', 'alldeads', 'newdeath', 'allrecived', 'activecases', 'activecases',
'critical', 'cases/1m', 'death/1m', 'alltest', 'test1/m', 'population', 'continent', 'e']
    soup = BeautifulSoup(source, 'lxml')
    got = "
    golf = 0
    bawara = "
    czas = time.time() * 1000
    czas = round(czas, 0)
    czas = int(czas)
    czas = str(czas)
```

Na początku zdefiniowaliśmy sobie listy oraz kategorie na stronie. Następnie wybraliśmy stronę z której bierzemy informacje. Wyznaczamy zmienne czas – żeby nasza aplikacja wiedziała z kiedy są te dane oraz czy ma je aktualizować.

Autorzy:  
Patryk Migaj  
Marcin Kryjom

```
for article in soup.find_all('tbody'):
    for zmienna in article.find_all('tr'):
        golf += 1
        a = -1
        yu = 0
        for zmienna2 in zmienna.find_all('td'):
            krzeslo = 0
            if a == -1:
                a += 1
                continue
            if a == 15:
                continue
            if a == 0:
                kraj = zmienna2

            kraj = kraj.text.replace('</nobr>', '').replace('\n', '')
            a += 1
        else:
            b = len(zmienna.find_all('td'))
            if not kraj in dzialy:
                if a == 6:
                    a += 1
                    yu += 1
                    continue
                grabowska = kraj + ';' + czas + ';' + list[6] + ';' + zmienna2.text.replace('\n', '').replace(',', '').replace(' ', '')
                got += '[]/info;'
                got += grabowska + '\n'
            else:
                bawara = kraj + ';' + czas + ';' + list[a] + ';' + zmienna2.text.replace('\n', '').replace(',', '').replace(' ', '')
                bawara = bawara.split(';')
                if bawara[3] == "":
                    bawara[3] = 'null'
                bawara = ';'.join(bawara)

                got += '[]/info;'
                bawara = bawara.replace('N/A', 'null')

                got += bawara + '\n'

        else:
            continue

        a += 1
        yu += 1
    return got
```

Tworzymy pętlę która nam po kolei wybiera informacje z tabeli

Następnie tworzymy kolejną pętlę która wybiera nam już szczególne dane, tworzymy zmienne które są odpowiedzialne za poprawną działalność pętli oraz kategori

Pozostaje nam tylko sformatować tekst za pomocą funkcji .replace i gotowe. Tak wygląda output.

```
[/info;Maldives;1591274458701;activecases;1199
[/info;Maldives;1591274458701;critical;9
[/info;Maldives;1591274458701;cases/1m;3427
[/info;Maldives;1591274458701;death/1m;13
[/info;Maldives;1591274458701;alltest;11775
[/info;Maldives;1591274458701;test1/m;21814
```

Autorzy:

Patryk Migaj

Marcin Kryjom

Serwer jest napisany w taki sposób że wysyła tylko i wyłącznie informację o dane o kraju który wpisał użytkownik:

```
if '/info;' in msg:
    warszawa = ""
    poznan = 0
    jawa = Html.funkcja().split('[]')
    msg = msg.replace('\r\n', '')
    for d in jawa:
        if poznan == 13 :
            break
        else:
            if msg in d:
                warszawa += d
                warszawa += '\n'
                poznan += 1
    self.csocket.send(warszawa.encode('UTF-8'))
    warszawa = ""
```

Następnie gdy Aplikacja dostanie informacje wpisuje odpowiednie wartości w odpowiednie rubryczki.

```
@SuppressWarnings("HandlerLeak") final Handler handler = handleMessage(msg) + {
    switch (msg.getData().getString( key: "key")) {
        case "allcases":
            updateTextView(R.id.stats_text_allcases_vol, msg.getData().getString( key: "value"));
            break;
        case "newcases":
            updateTextView(R.id.stats_text_newcases_vol, msg.getData().getString( key: "value"));
            break;
        case "alldeads":
            updateTextView(R.id.stats_text_alldeads_vol, msg.getData().getString( key: "value"));
            break;
        case "newdeath":
            updateTextView(R.id.stats_text_newdeath_vol, msg.getData().getString( key: "value"));
            break;
        case "allrecived":
            updateTextView(R.id.stats_text_allrecived_vol, msg.getData().getString( key: "value"));
            break;
        case "activecases":
            updateTextView(R.id.stats_text_activecases_vol, msg.getData().getString( key: "value"));
            break;
        case "critical":
            updateTextView(R.id.stats_text_critical_vol, msg.getData().getString( key: "value"));
            break;
        case "cases/in":
            updateTextView(R.id.stats_text_casesin_vol, msg.getData().getString( key: "value"));
            break;
        case "death/in":
            updateTextView(R.id.stats_text_deathin_vol, msg.getData().getString( key: "value"));
            break;
        case "alltest":
            updateTextView(R.id.stats_text_alltest_vol, msg.getData().getString( key: "value"));
            break;
        case "test1/a":
            updateTextView(R.id.stats_text_test1a_vol, msg.getData().getString( key: "value"));
            break;
        case "population":
            updateTextView(R.id.stats_text_population_vol, msg.getData().getString( key: "value"));
            break;
        case "continent":
            updateTextView(R.id.stats_text_continent_vol, msg.getData().getString( key: "value"));
            break;
        default:
            Utils.Log("missing", "Missing implementation of " + msg.getData().getString( key: "key") + "");
            break;
    }
};
```

Autorzy:  
Patryk Migaj  
Marcin Kryjom  
Wygląd końcowy:

## ← Statystyki

Poland

Kraj:	Poland
Wszystkie zarażenia:	26561
Nowe zarażenia:	575
Wszystkie zgony:	1157
Nowe zgony:	4
Wyleczonych:	12855
Aktywne zakażenia:	12549
Poważne przypadki:	160
Chorych na 1 milion:	702
Zgonów na 1 milion:	31
Wszystkie testy:	1038281
Testy na 1 mln:	27432
Populacja:	37849184
Kontynent:	Europe

---

Autorzy:

Patryk Migaj

Marcin Kryjom



Autorzy:  
Patryk Migaj  
Marcin Kryjom

## - Aktualne zarażenia

Jest to komenda wykorzystywana do podania informacji o aktualnych zarażeniach na świecie na stronie głównej aplikacji. Przy starcie programu ładowana jest informacja z bazy danych a następnie wysyłana jest komenda /zarażenia:

```
private void loadAllCasesMain(){
    final TextView textView = findViewById(R.id.home_text_allcases);
    final String a = "Zachorowań na świecie: %s";
    final String cases = DatabaseHelper.getValue( context: this, key: "allcases");
    if(cases != null){
        textView.setText(String.format(a, cases));
    }

    @SuppressWarnings("HandlerLeak") final Handler handler = handleMessage(msg) -> {
        textView.setText(String.format(a,msg.getData().getString( key: "text")));
    };
    DataManager.getDataSocketConnection().putCallback( key: "/zarażenia", (key, value) -> {
        Message msg = handler.obtainMessage();
        Bundle bundle = new Bundle();
        bundle.putString("text",value);
        msg.setData(bundle);
        handler.sendMessage(msg);
        DatabaseHelper.setValue( context: MainActivity.this, key: "allcases",value);
    });
    DataManager.getDataSocketConnection().putCallback( key: "stateupdate", (key, value) -> {
        if(value.equals("OPEN")){
            DataManager.getDataSocketConnection().send( cmd: "/zarażenia");
            DataManager.getDataSocketConnection().removeCallback(key, callback: this);
        }
    });
}
```

Serwer łąduje funkcje:

Oraz wysyłana jest informacja

```
def funkcja():
    lista_kraje= []
    source = requests.get('https://www.worldometers.info/coronavirus/').text
    soup = BeautifulSoup(source, 'xml')
    match = soup.find('div', class_='maincounter-number')
    match = match.text
    match = match.replace('\n', '/zarażenia;', 1)

    return match
```

Output:

```
/zarażenia
[OUT] /zarażenia
[Svr-IN] /zarażenia;6,876,347
```

Autorzy:

Patryk Migaj

Marcin Kryjom

Gdy Aplikacja otrzyma prawidłową informację wkleja ją do odpowiedniego miejsca(kod wyżej)