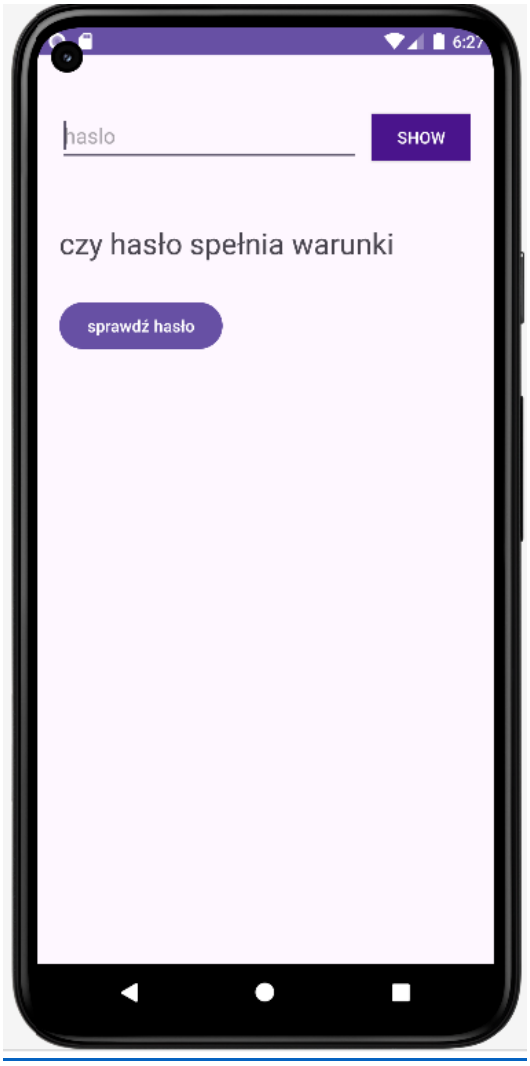
Zadanie 3.

Wykonaj aplikację mobilną za pomocą środowiska programistycznego dostępnego na stanowisku egzaminacyjnym oraz uruchom ją w dostępnym emulatorze systemu mobilnego.



Aplikacja Androida sprawdza poprawność hasła pod względem określonych warunków.

Aplikacja powinna zawierać interfejs użytkownika składający się z pola tekstowego do wprowadzenia hasła, przycisku do jego weryfikacji oraz przełącznika umożliwiającego wyświetlenie hasła w formie jawnej.

Warunki poprawności hasła są następujące:

- Hasło musi mieć co najmniej 8 znaków.
- Hasło musi zawierać co najmniej jedną dużą literę.

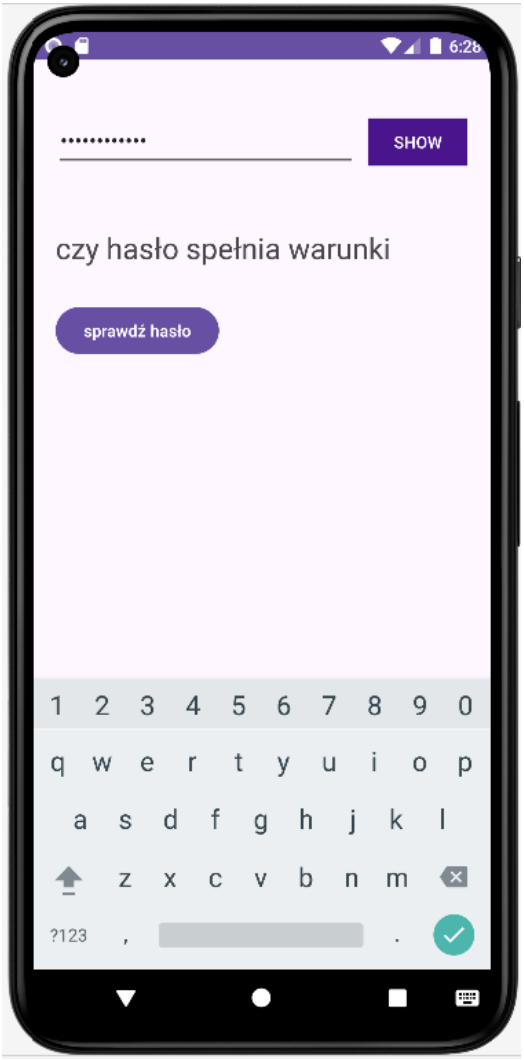
- Hasło musi zawierać co najmniej jedną cyfrę.
- Hasło musi zawierać co najmniej jeden znak specjalny spośród: !@#$%^&*()-+=.

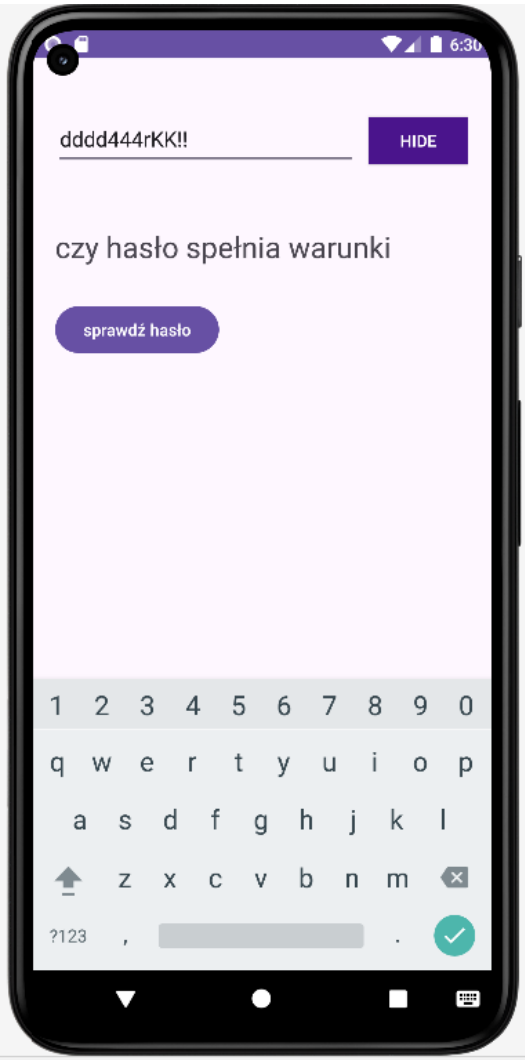
W przypadku poprawnego hasła, wyświetl komunikat "Hasło spełnia warunki", w przeciwnym razie wyświetl komunikat "Hasło nie spełnia warunków".

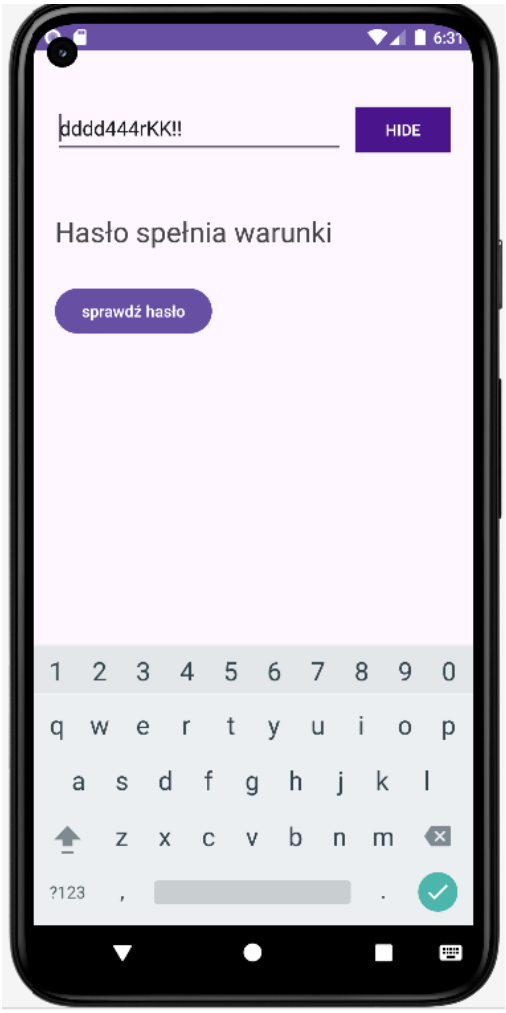
Dodatkowo, aplikacja powinna umożliwiać pokazywanie i ukrywanie wprowadzanego hasła za pomocą przełącznika.

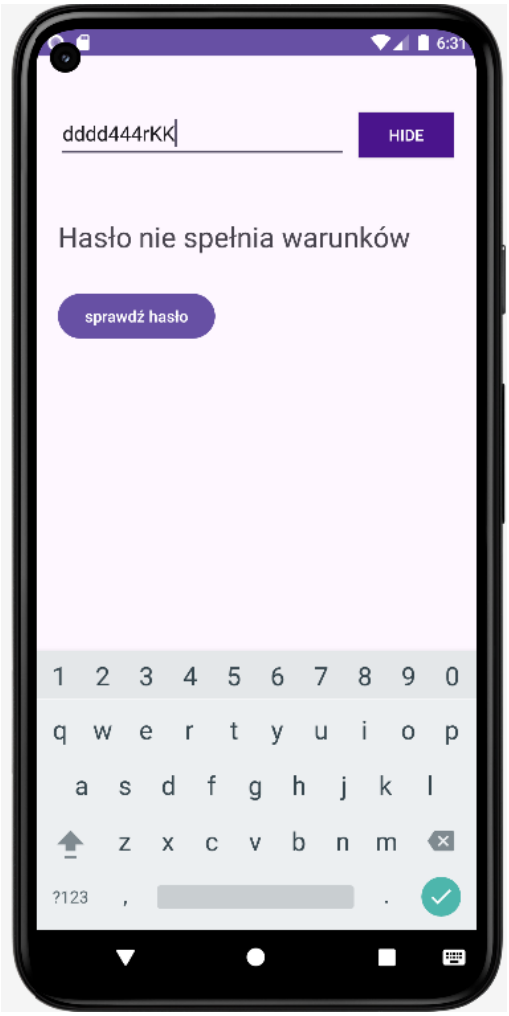
Opis wyglądu aplikacji :

1. Pole Hasła :
    - Ustaw tekst pomocniczy na "Wprowadź hasło".
    - Ustaw typ wprowadzania na textPassword dla ukrycia wpisywanych znaków.
    - Ustaw minimalną wysokość na 48dp, aby zachować odpowiednią przestrzeń dla wpisywanych danych.
    - Ustaw wagę na 3, aby pole zajmowało większą część dostępnego miejsca.
    - Opcjonalnie możesz ustawić podpowiedzi do automatycznego uzupełniania.
2. Przełącznik Wyświetlania Hasła (ToggleButton):
    - Ustaw kolor tła na jasnofioletowy (#4A148C).
    - Ustaw tekst na "Pokaż/ukryj".
    - Ustaw kolor tekstu na biały.
    - Ustaw teksty dla stanów wyłączonego i włączonego.
3. Przycisk Sprawdzenia Hasła (Button):
    - Ustaw tekst na "Sprawdź hasło".
4. Komunikat Wyniku Sprawdzenia (TextView):
    - Ustaw początkowy tekst na "Czy hasło spełnia warunki?".
    - Ustaw rozmiar tekstu na 25sp.


6:28
SHOW
czy hasło spełnia warunki
sprawdź hasło

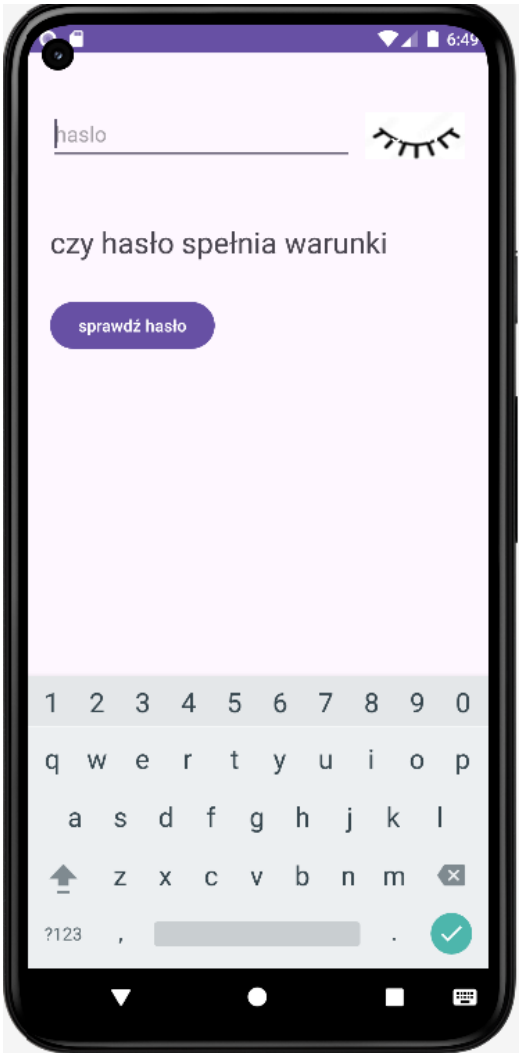

6:30
dddd444rKK!!
HIDE
czy hasło spełnia warunki
sprawdź hasło

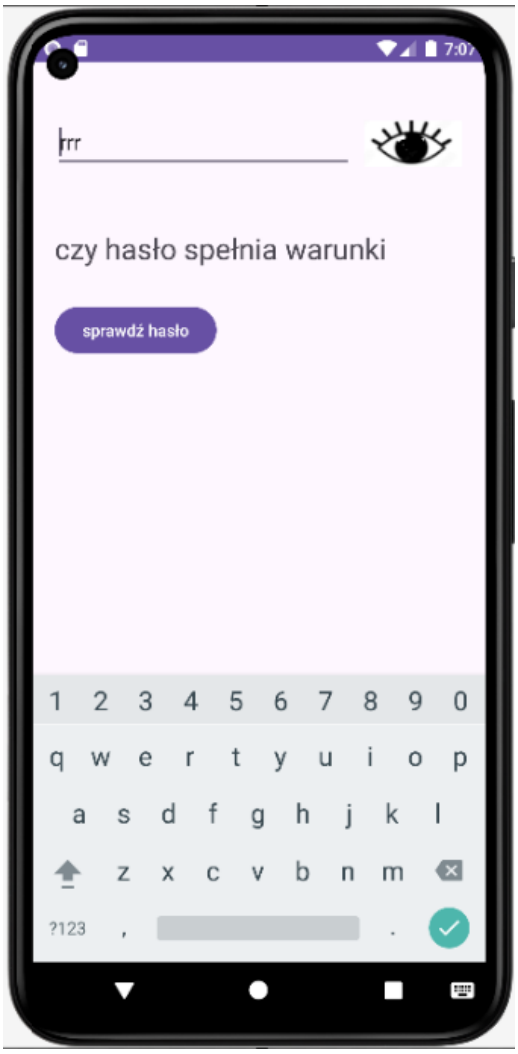




Testowanie Aplikacji:

- Przetestuj aplikację, upewniając się, że wszystkie funkcje działają zgodnie z oczekiwaniami.
- Zweryfikuj, czy komunikaty wyświetlane użytkownikowi odpowiadają poprawnie zweryfikowanym hasłom oraz tym, które nie spełniają wymagań.





Dodatkowo obsługa Przełącznika Wyświetlania Hasła:

Ustawienie tła przełącznika na odpowiednią grafikę (otwarte lub zamknięte oczko) w zależności od kliknięcia ToggleButton. Rys. powyżej.

```
showPass.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(isChecked) {
            pass.setTransformationMethod(null);
            //showPass.setBackgroundDrawable(getDrawable(oczko_otwarte));
        }
        else
        {
            pass.setTransformationMethod(PasswordTransformationMethod.getInstance());
            //showPass.setBackgroundDrawable(getDrawable(oczko_zamkniete));
        }
    }
});
```

```
private boolean sprawdzHaslo(String haslo)
{
    String regex = "^(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*()-+=]).{8,}$";
    // Minimum 8 znaków, przynajmniej 1 duża litera, 1 cyfra, 1 znak specjalny

    //  tworzy obiekt klasy Pattern, która reprezentuje skompilowane wyrażenie regularne.Wyrażenie
    //  regularne jest określone przez zmienną regex, która jest przekazana do metody compile().
    Pattern pattern = Pattern.compile(regex);

    //tworzy obiekt klasy Matcher, który będzie używany do
    // dopasowywania wyrażenia regularnego do ciągu znaków.
    Matcher matcher = pattern.matcher(haslo);

    return matcher.matches();
}
```

```
private boolean spr(String haslo)
{
    int ileC = 0, ileDL=0, ileL=0, ileB=0;

    for (int i = 0; i < haslo.length(); i++) {
        if (Character.isDigit(haslo.charAt(i))) ileC++;          // cyfra
        if (Character.isUpperCase(haslo.charAt(i))) ileDL++;     //duża litera
        if (Character.isLetter(haslo.charAt(i))) ileL++;         // litera
        if (Character.isWhitespace(haslo.charAt(i))) ileB++;     //znak biały

    }

    int ileZ = haslo.length()-ileC-ileL-ileB;                    // znak specjalny

    if(ileC!=0 && ileZ!=0 && ileDL!=0 && ileB==0 && haslo.length()>=8) return true;
    else return false;
}
```

Zadanie 4.

Twoim zadaniem jest zaprojektowanie i implementacja aplikacji mobilnej, która umożliwi użytkownikom sprawdzanie różnych warunków dotyczących wprowadzonych danych. Aplikacja powinna być prostym narzędziem do weryfikacji danych użytkownika pod kątem określonych kryteriów.

Ogólne wytyczne aplikacji:

1. Ekran główny:
    - Interfejs aplikacji powinien być oparty o układ Linearny z zagnieżdżonym układem Tabelarycznym z 2 kolumnami.
    - Na górze ekranu powinno znajdować się pole tekstowe do wprowadzania danych (np. hasło, kod, numer).
    - Poniżej button-y i pola tekstowe w poszczególnych wierszach.
2. Warunki sprawdzane:
    - Dla każdego warunku powinien być przycisk, który po naciśnięciu uruchomi sprawdzenie.
    - Po prawej stronie przycisku powinien być wyświetlany wynik sprawdzenia (np. "Tak" lub "Nie").

Szczegółowe wytyczne wyglądu aplikacji:

Interfejs aplikacji :

- powinien być ułożony w pionową orientację.
- Odległość od krawędzi ekranu powinna wynosić 10dp.
- Na górze ekranu znajduje się pole tekstowe przeznaczone do wprowadzenia danych, o pełnej szerokości ekranu.
- Odległość między polem tekstowym a pierwszym wierszem przycisku wynosi 25dp.
- Każdy przycisk wraz z odpowiadającym mu tekstem powinien być umieszczony w osobnym wierszu.

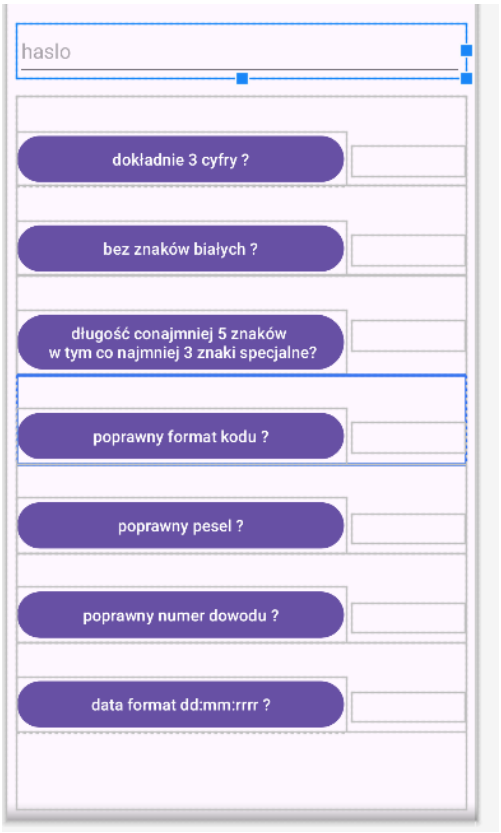
Pola tekstowe i przyciski:

- Przyciski mają stałą szerokość i wysokość, aby były odpowiednio dostosowane do treści.
- Każdy przycisk powinien mieć margines górny wynoszący 30dp.
- Teksty powinny być czytelne i konkretnie opisywać warunek, np. "dokładnie 3 cyfry ?", "bez znaków białych ?", itp.
- Odległość między przyciskiem a odpowiadającym mu polem tekstowym wynosi 35dp.
- Teksty wyświetlające wyniki sprawdzeń mają rozmiar 20sp.
- Margines lewy między przyciskiem a odpowiadającym mu tekstem wynosi 5dp.

Kontrolki wiersza:

- Każdy wiersz w tabeli (TableRow) powinien mieć wysokość zawierającą się w zawartości, zapewniając optymalne ułożenie elementów.
- Przyciski i teksty wynikowe powinny być umieszczone w odpowiednich kolumnach (layout_column="0" dla przycisków, layout_column="1" dla tekstów wynikowych).

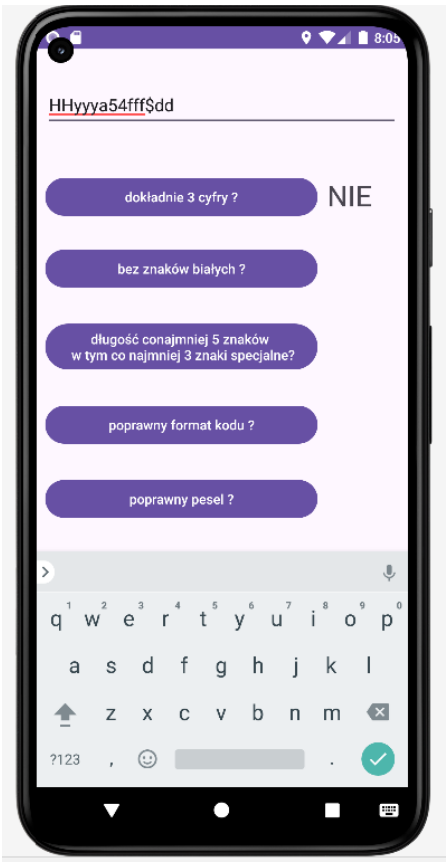
Rys. poglądowy

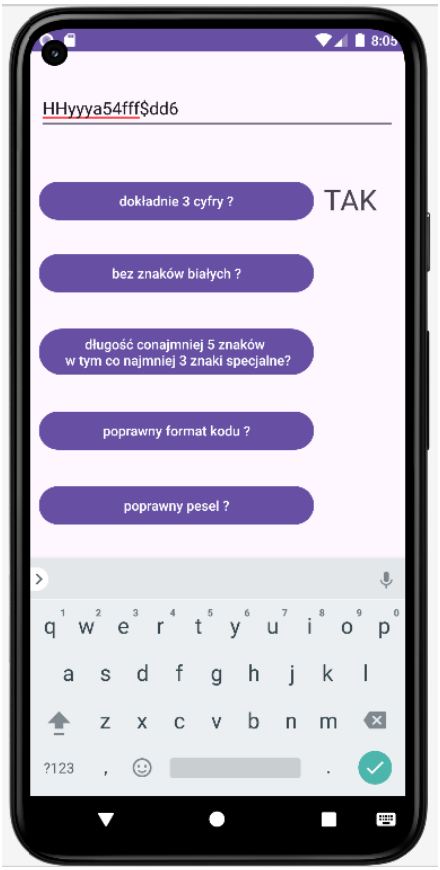


Zadania do wykonania na podstawie kodu:

1. Implementacja Sprawdzania Cyfr w Haśle:
    - Zaimplementuj przycisk btn1 tak, aby po jego kliknięciu sprawdzał, czy hasło zawiera dokładnie 3 cyfry.
    - W przypadku spełnienia warunku, wyświetl "TAK" w przeciwnym razie "NIE".
2. Implementacja Sprawdzania Znaków Białych:
    - Zaimplementuj przycisk btn2 tak, aby po jego kliknięciu sprawdzał, czy hasło nie zawiera znaków białych.
    - W przypadku spełnienia warunku, wyświetl "TAK" w przeciwnym razie "NIE".
3. Implementacja Sprawdzania Długości i Znaków Specjalnych:
    - Zaimplementuj przycisk btn3 tak, aby po jego kliknięciu sprawdzał, czy hasło zawiera co najmniej 3 znaki specjalne i ma długość co najmniej 5 znaków.
    - W przypadku spełnienia warunku, wyświetl "TAK" na txt3, w przeciwnym razie "NIE".
4. Implementacja sprawdzania Formatu Kodu:
    - Zaimplementuj przycisk btn4 tak, aby po jego kliknięciu sprawdzał, czy wprowadzony kod ma poprawny format.
    - W przypadku spełnienia warunku, wyświetl "TAK" w przeciwnym razie "NIE".
5. Implementacja sprawdzania numeru PESEL:
    - Zaimplementuj przycisk btn5 tak, aby po jego kliknięciu sprawdzał, czy wprowadzony numer PESEL posiada tylko cyfry i ma dokładnie 11 znaków.
    - W przypadku spełnienia warunku, wyświetl "TAK" w przeciwnym razie "NIE".
6. Implementacja sprawdzania numeru Dowodu Osobistego:
    - Zaimplementuj przycisk btn6 tak, aby po jego kliknięciu sprawdzał, czy wprowadzony numer dowodu składa się z 3 dużych liter i 4 cyfr

- W przypadku spełnienia warunku, wyświetl "TAK" w przeciwnym razie "NIE".





Wspólne style do Button i TextView :

activity_main.xml | style.xml | MainActivity.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<style name="button">
    <item name="android:layout_width">wrap_content</item>
     <item name="android:layout_height">wrap_content</item>
     <item name="android:layout_column">0</item>
     <item name="android:layout_marginTop">30dp</item>
</style>
<style name="Text">
     <item name="android:layout_width">wrap_content</item>
     <item name="android:layout_height">wrap_content</item>
     <item name="android:layout_marginTop">35dp</item>
     <item name="android:layout_weight">2</item>
     <item name="android:layout_marginLeft">10dp</item>
     <item name="android:textSize">30sp</item>
     <item name="android:layout_column">1</item>
     <item name="android:text"></item>
</style>
</resources>
```

```java
    btn1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String haslo1 = haslo.getText().toString().trim();
            if (cyfry_3(haslo1) == 3) txt1.setText("TAK");
            else txt1.setText("NIE");
        }
    });



    private int cyfry_3(String haslo) {
//          String regex = ".*[0-9].*[0-9].*[0-9].*";
//          Pattern patern = Pattern.compile(regex);
//          Matcher matcher = patern.matcher(haslo);
//          return matcher.matches();
        int d = 0;
        for (int i = 0; i < haslo.length(); i++)
            if (Character.isDigit(haslo.charAt(i))) d++;
        return d;
    }
```