|  |  |  |  |
| --- | --- | --- | --- |
|  | Politechnika Bydgoska im. J. J. Śniadeckich  Wydział Telekomunikacji,  Informatyki i Elektrotechniki | |  |
| **Przedmiot** | Programowanie urządzeń mobilnych | | |
| **Prowadzący** | dr inż. Mirosław Miciak | | |
| **Temat** | Lab 2 Stan aplikacji | | |
| **Student** | Cezary Tytko | | |
| **Nr lab.** | 2 | **Data wykonania** | 17.03.2025 |
| **Ocena** |  | **Data oddania spr.** |  |

1. Aplikacja

class MainActivity : AppCompatActivity() {

private val TAG = "MainActivity

private val PREFS\_NAME = "counterPrefs" // Nazwa pliku z preferencjami

private val COUNTER\_KEY = "clickCounter" // Klucz, pod którym zapisujemy licznik

// Zmienna przechowująca stan licznika

private var clickCount = 0

// Zapisywanie stanu licznika przy zmianie orientacji lub zamknięciu aplikacji

override fun onSaveInstanceState(outState: Bundle) {

super.onSaveInstanceState(outState)

outState.putInt(COUNTER\_KEY, clickCount)

logAndAppend("onSaveInstanceState called")

}

// Przywracanie stanu licznika po zmianie orientacji lub ponownym otwarciu aplikacji

override fun onRestoreInstanceState(savedInstanceState: Bundle) {

super.onRestoreInstanceState(savedInstanceState)

clickCount = savedInstanceState.getInt(COUNTER\_KEY, 0)

logAndAppend("onRestoreInstanceState called")

}

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

logAndAppend("onCreate called")

setContentView(R.layout.activity\_main)

// Przywrócenie stanu licznika po przywróceniu aktywności

if (savedInstanceState != null) {

clickCount = savedInstanceState.getInt(COUNTER\_KEY, 0)

}

else{ // Przywrócenie licznika z zapisanych preferencji po zamknięciu aplikacji

val sharedPreferences = getSharedPreferences(PREFS\_NAME, Context.MODE\_PRIVATE)

clickCount = sharedPreferences.getInt(COUNTER\_KEY, 0) // Domyślnie 0, jeśli brak danych

logAndAppend("sharedPreferences readed")

}

val textViewCounter: TextView = findViewById(R.id.textViewCounter)

val buttonClick: Button = findViewById(R.id.buttonClick)

// Ustawienie początkowej wartości licznika

textViewCounter.text = clickCount.toString()

// Obsługa kliknięcia przycisku

buttonClick.setOnClickListener {

clickCount++ // Zwiększ liczbę kliknięć

textViewCounter.text = clickCount.toString() // Zaktualizuj widok

}

}

override fun onStart() {

super.onStart()

logAndAppend("onStart called")

}

override fun onResume() {

super.onResume()

logAndAppend("onResume called")

}

override fun onPause() {

super.onPause()

logAndAppend("onPause called")

}

override fun onStop() {

super.onStop()

logAndAppend("onStop called")

}

override fun onRestart() {

super.onRestart()

logAndAppend("onRestart called")

}

override fun onDestroy() {

super.onDestroy()

logAndAppend("onDestroy called")

val sharedPreferences = getSharedPreferences(PREFS\_NAME, Context.MODE\_PRIVATE)

with(sharedPreferences.edit()) {

putInt(COUNTER\_KEY, clickCount) // Zapisz stan licznika

apply()

}

}

private fun logAndAppend(message: String) {

Log.d(TAG, message)

}

}

Aplikacja wykorzystuje zarówno stan zapisany w preferencjach aplikacji zapisywanych w pamięci urządzenia mobilnego co pozwala persystować dane w przypadku wyłączenia aplikacji i ładować je przy jej ponownym uruchomieniu, jak i korzysta ze stanu instancji która przechowuje informacje w ramach raz uruchomionej aplikacji miedzy jej stanami np. w przypadku zminimalizowania i wybudzenia odzyskamy zachowane w ten sposób informacje.

1. Prezentacja
2. Wnioski

Ćwiczenie nie było trudne dzięki dobrze opisanemu przykładowi użyciu stanów aplikacji, oraz przeciążania odpowiednich zdarzeń jak onCreate, OnDestroy w instrukcji do laboratorium, co pozwoliło łatwo wykonać zadanie stworzenia licznika przechowującego swój stan niezależnie od okoliczności jak zamkniecie, oraz minimalizacja aplikacji.