Zespół Szkół NR.10
Im. Stanisława Staszica

Sprawozdanie z projektu
ThrowMe

Autor: Adam Płochocki
Klasa: 4AP

Warszawa, 17.09.2025 r.

Spis treści

Cel projektu

Celem projektu jest stworzenie aplikacji mobilnej na system Android, która umożliwia:

- Rejestrację i logowanie użytkowników
- Wykonywanie pomiarów rzutów telefonu z wykorzystaniem czujników
- Wyświetlanie tabeli wyników wszystkich użytkowników
- Przechowywanie i wyświetlanie historii pomiarów użytkownika

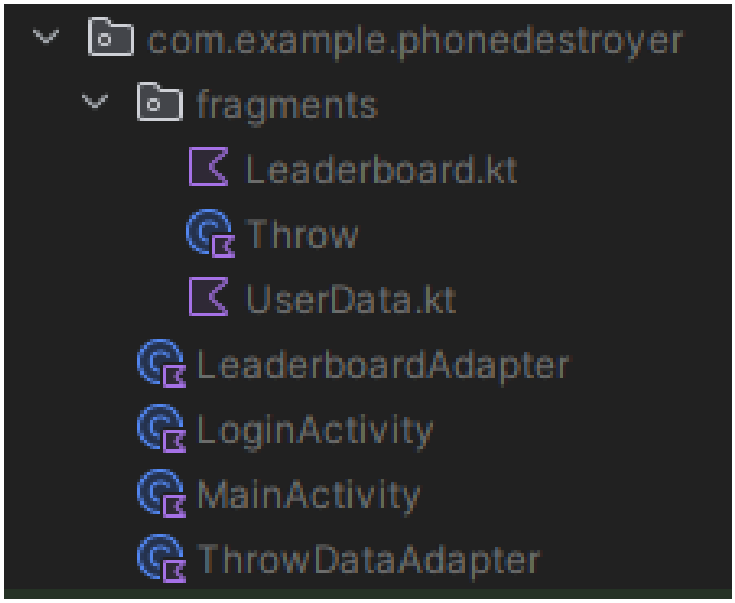
Główne komponenty

Activities:

- MainActivity - zarządza nawigacją między fragmentami
- LoginActivity - obsługuje autoryzację użytkowników

Fragments:

- Throw - interfejs do wykonywania pomiarów
- Leaderboard - wyświetla ranking graczy
- UserData - pokazuje dane i historię użytkownika



Implementacja funkcjonalności

System Logowania

LoginActivity.kt implementuje system autoryzacji użytkowników z wykorzystaniem SharedPreferences do przechowywania danych sesji.

```
private fun validate(): Pair<String, String>{
    val usr = usernameInput.text.toString()
    val pass = passwordInput.text.toString()

    if(usr.isNullOrEmpty() || pass.isNullOrEmpty()) throw Error("Nazwa użytkownika i hasło nie mogą być puste")

    return Pair(usr, pass)
}

// TODO : remove the default uid when api is done
private fun proceed(username: String, uid: String = "019ac191-dcec-7968-a6e8-ff3dfa85926e"){
    val sp = getSharedPreferences("user_data", MODE_PRIVATE);

    sp.edit {
        putString("username", username)
        putString("uid", uid)
    }

    val intent = Intent(this, MainActivity::class.java)
    startActivity(intent)
    finish()
}
```

Główna nawigacja

MainActivity zarządza systemem nawigacji typu "tab bar" w stylu z trzema głównymi sekcjami.

```kotlin
private fun setActiveTab(activeButton: Button) {
    // Set all buttons to inactive first
    allButtons.forEach { button ->
        button.setTextColor(Color.parseColor("#8E8E93"))
        button.setTypeface(null, android.graphics.Typeface.NORMAL)
        button.foreground = null
    }

    // Set the clicked button to active
    activeButton.setTextColor(Color.BLACK)
    activeButton.setTypeface(null, android.graphics.Typeface.BOLD)
    activeButton.foreground = createUnderlineDrawable()
}
```

Funkcjonalność:

- Dynamiczne podkreślanie aktywnego taba
- Automatyczne dezaktywowanie pozostałych tabów
- Wymiana fragmentów w kontenerze
- Sprawdzanie statusu logowania użytkownika

Fragment widoku rzutu (Throw.kt)

```kotlin
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    resultText = view.findViewById(R.id.resultText)
    startButton = view.findViewById(R.id.startButton)

    sensorManager = requireActivity().getSystemService(Context.SENSOR_SERVICE) as SensorManager

    val accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION)
    val gyro = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE)

    sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_GAME)
    sensorManager.registerListener(this, gyro, SensorManager.SENSOR_DELAY_GAME)

    startButton.setOnClickListener {
        accelWindow.clear()
        recording = true
        windowActive = false
        currentPitch = 0.0
        lastGyroTimestamp = 0L
        resultText.text = "Rzuć telefonem !"
    }
}
```

Implementacja:

- Wykorzystanie akcelerometru i żyroskopu
- Detekcja momentu rzutu
- Obliczanie maksymalnego przyspieszenia i prędkości
- Wysyłanie danych do API

Tabela wyników

Fragment wyświetlający ranking wszystkich użytkowników aplikacji.

```
private fun loadLeaderboardData() {
    progressBar.visibility = View.VISIBLE
    errorText.visibility = View.GONE
    recyclerView.visibility = View.GONE

    CoroutineScope(Dispatchers.IO).launch {
        try {
            val apiUrl = "" // TODO :  make the api and add its url
            val response = URL(apiUrl).readText()
            val jsonArray = JSONArray(response)

            val entries = mutableListOf<LeaderboardEntry>()
            for (i in 0 ≤  until  < jsonArray.length()) {
                val item = jsonArray.getJSONObject(i)
                entries.add(
                    LeaderboardEntry(
                        position = i + 1,
                        playerName = item.getString("username"),
                        score = item.getInt("score")
                    )
                )
            }

            withContext(Dispatchers.Main) {
                adapter.submitList(entries)
                progressBar.visibility = View.GONE
                recyclerView.visibility = View.VISIBLE
            }

        } catch (e: Exception) {
            withContext(Dispatchers.Main) {
                progressBar.visibility = View.GONE
                errorText.visibility = View.VISIBLE
                errorText.text = "Nie udało się załadować tabeli wyników"
            }
        }
    }
}
```

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    val entry = entries[position]

    holder.playerName.text = entry.playerName
    holder.score.text = "${entry.score} pkt"

    // special styling for top 3
    when (entry.position) {
        1 -> {
            holder.medal.visibility = View.VISIBLE
            holder.medal.text = "🥇"
            holder.position.visibility = View.GONE
            holder.card.setCardBackgroundColor(Color.parseColor("#FFF9E6"))
        }
        2 -> {
            holder.medal.visibility = View.VISIBLE
            holder.medal.text = "🥈"
            holder.position.visibility = View.GONE
            holder.card.setCardBackgroundColor(Color.parseColor("#F5F5F5"))
        }
        3 -> {
            holder.medal.visibility = View.VISIBLE
            holder.medal.text = "🥉"
            holder.position.visibility = View.GONE
            holder.card.setCardBackgroundColor(Color.parseColor("#FFF5E6"))
        }
        else -> {
            holder.medal.visibility = View.GONE
            holder.position.visibility = View.VISIBLE
            holder.position.text = "${entry.position}"
            holder.card.setCardBackgroundColor(Color.WHITE)
        }
    }
}
```

Funkcjonalność:

- Pobieranie danych z API
- Wyświetlanie top 3 z medalami
- Specjalne tło dla podium
- Efektywne przewijanie z RecyclerView

Dane użytkownika

Fragment przedstawiający statystyki i historię rzutów użytkownika.

```kotlin
private fun loadThrowData() {
    progressBar.visibility = View.VISIBLE
    errorText.visibility = View.GONE
    recyclerView.visibility = View.GONE

    CoroutineScope(Dispatchers.IO).launch {
        try {
            val sp = requireActivity().getSharedPreferences("user_data", Context.MODE_PRIVATE)
            val uid = sp.getString("uid", "")
            val apiUrl = ""
            val response = URL(apiUrl).readText() // TODO : make the api and put the url here
            val jsonArray = JSONArray(response)

            val throws = mutableListOf<ThrowData>()
            for (i in 0 ≤  until  < jsonArray.length()) {
                val item = jsonArray.getJSONObject(i)
                throws.add(
                    ThrowData(
                        distance = item.getString("distance"),
                        peakAcceleration = item.getString("peak_acceleration"),
                        peakSpeed = item.getString("peak_speed")
                    )
                )
            }

            withContext(Dispatchers.Main) {
                adapter.submitList(throws)
                progressBar.visibility = View.GONE
                recyclerView.visibility = View.VISIBLE
            }

        } catch (e: Exception) {
            withContext(Dispatchers.Main) {
                progressBar.visibility = View.GONE
                errorText.visibility = View.VISIBLE
                errorText.text = "Nie udało się załadować danych"
            }
```

Funkcjonalność:

- Wyświetlanie nazwy użytkownika
- Lista wszystkich rzutów z danymi (Dystans, Max. Przyspieszenie, Prędkosc)
- Tabela z nagłówkami kolumn
- Przycisk wylogowania

Możliwości rozwoju

Planowane Funkcjonalności

- Animacje przejść między fragmentami
- Przejście między fragmentami po przesunięciu palca
- Wykres postępów użytkownika
- System osiągnięć i nagród
- Udostępnianie wyników w social media
- Tryb ciemny (Dark Mode)

Wykorzystane biblioteki

'androidx.core:core-ktx:1.12.0'

'androidx.appcompat:appcompat:1.6.1'

'com.google.android.material:material:1.9.0'

'androidx.constraintlayout:constraintlayout:2.1.4'

'androidx.recyclerview:recyclerview:1.3.2'

'androidx.cardview:cardview:1.0.0'

'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.3'