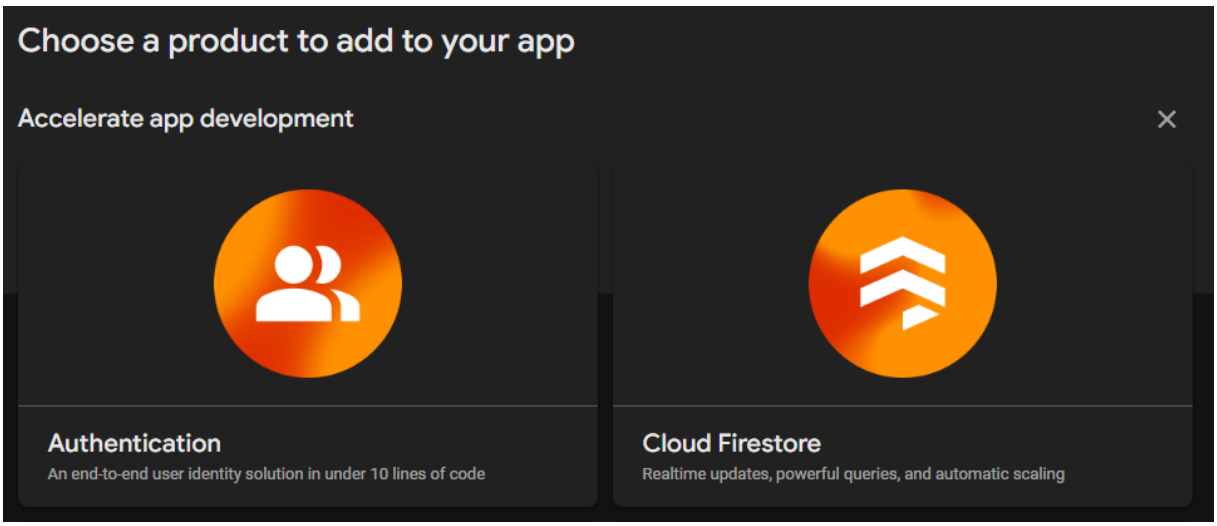
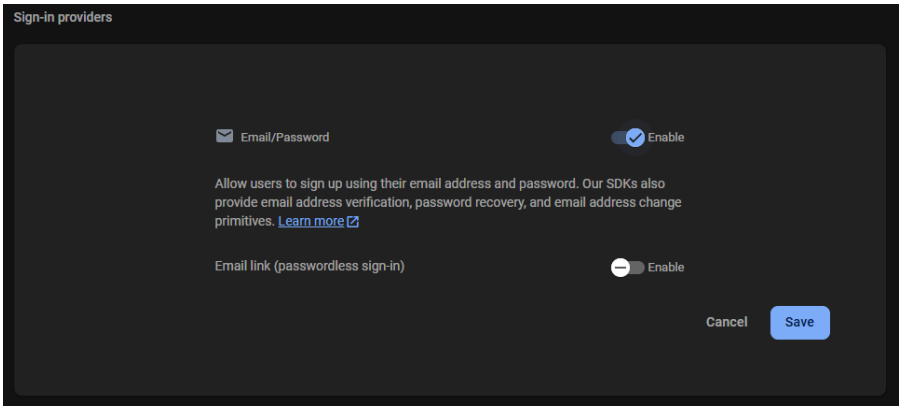
Czujniki, Firebase c.d – Logowanie, Rejestracja,

Podobnie jak na poprzednich zajęciach Tworzymy aplikację w android studio oraz w konsoli firebase.

Dodatkowo poza firestore włączamy również autentykację:

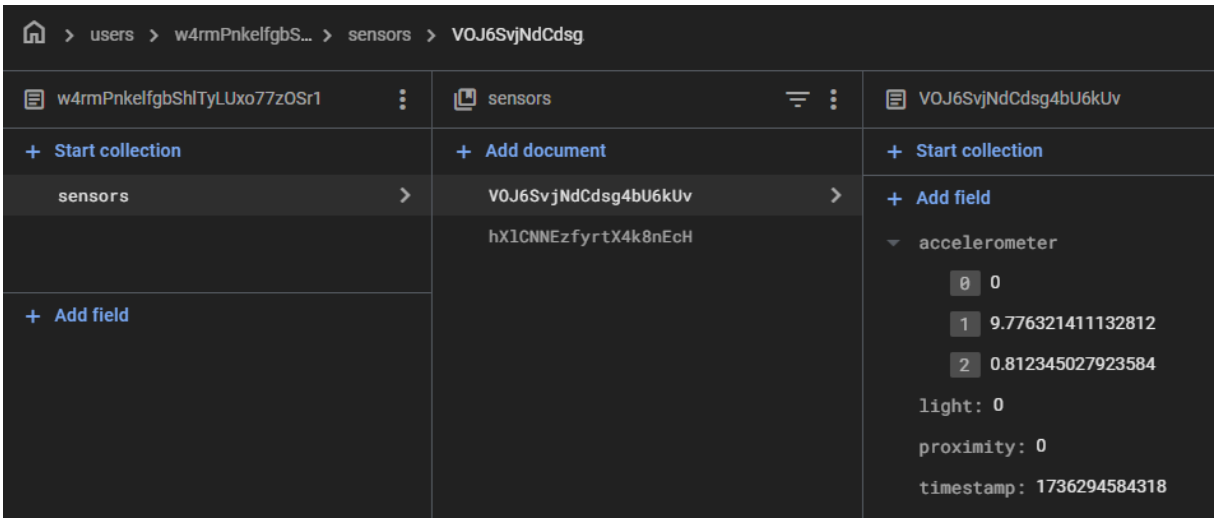




W bazie będziemy przechowywać dane z czujników, zależy nam na tym aby tylko zalogowani użytkownicy mogli zapisywać i odczytywać dane. Dodatkowo każdy użytkownik powinien mieć dostęp wyłącznie do swoich danych. Pomoże w tym odpowiednie ustawienie reguł bazy:

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId}/sensors/{sensorDocId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }
  }
}
```

Dane będą przechowywane w dokumencie userId w kolekcji sensors w dokumencie sensorDocId:



Z włączoną autentykacją w firebase, w aplikacji androidowej wystarczy dodać odpowiednie zależności:

```
implementation(platform("com.google.firebase:firebase-bom:33.7.0"))
implementation ("com.google.firebase:firebase-auth-ktx")
implementation ("com.google.firebase:firebase-firestore-ktx")
```

Używamy instancji Firebase Authentication do logowania/rejestracji

Oraz stworzyć formularz i dwa przyciski:

```
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Button
import androidx.compose.material3.OutlinedTextField

import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import com.google.firebase.auth.FirebaseAuth

@Composable
fun LogRegScreen(modifier: Modifier,onLoginSuccess: () -> Unit) {
    val auth = FirebaseAuth.getInstance()
    var email by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }

    Column(
        modifier = modifier
            .fillMaxSize()
```

```
            .padding(16.dp)
    ) {
        OutlinedTextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(8.dp))

        OutlinedTextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Haslo") },
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth()
        )

        Spacer(modifier = Modifier.height(16.dp))


        Button(
            onClick = {
                auth.signInWithEmailAndPassword(email, password)
                    .addOnCompleteListener { task ->
                        if (task.isSuccessful) {
                            onLoginSuccess()
                        } else {

                        }
                    }
            },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Zaloguj")
        }

        Spacer(modifier = Modifier.height(8.dp))


        Button(
            onClick = {
                auth.createUserWithEmailAndPassword(email, password)
                    .addOnCompleteListener { task ->
                        if (task.isSuccessful) {
                            onLoginSuccess()
                        } else {

                        }
                    }
            },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text("Zarejestruj")
        }
    }
}
```

Logowanie:

```
auth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener { task ->
        if (task.isSuccessful) {
            onLoginSuccess()
        } else {

        }
    }
```

Metoda signInWithEmailAndPassword pochodzi z interfejsu Firebase Authentication.

Próbuje ona zalogować użytkownika na podstawie podanego adresu e-mail i hasła.

Operacja ta jest asynchroniczna – nie blokuje głównego wątku aplikacji i uruchamia się w tle.

Analogicznie Rejestracja:

```
auth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                onLoginSuccess()
            } else {

            }
        }
}
```

Metoda .addOnCompleteListener { ... } rejestruje słuchacza (listener), który zostanie wywołany, gdy operacja logowania/rejestracji zakończy się (niezależnie od powodzenia).
Lambda { task -> ... } przyjmuje argument task, który reprezentuje wynik zakończonej operacji.
Funkcja onLoginSuccess zostaje zaimplementowana w innym miejscu aplikacji i przekazana do composable.

Pobranie danych z czujników oraz zapis do firebase jest również prostym zadaniem:

```
import android.content.Context
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import androidx.compose.foundation.layout.*
import androidx.compose.material3.Button
import androidx.compose.material3.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.ktx.firestore
import com.google.firebase.ktx.Firebase

@Composable
fun MainScreen(modifier: Modifier) {
    val context = LocalContext.current

//Manager sensorów z systemu:
    val sensorManager = remember {
```

```
        context.getSystemService(Context.SENSOR_SERVICE) as SensorManager
    }
//Wybrane czujniki:
       val accelerometer =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)

    val proximitySensor =
sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY)



 //Zmienne do przechowania danych z czujników accelerometr ma 3wartości dla
x,y,z:
    var accelerometerValues by remember { mutableStateOf(floatArrayOf(0f,
0f, 0f)) }
     var proximityValue by remember { mutableStateOf(0f) }


//Nasłuchiwacz zdarzeń dla czujników - aktualizacja zmiennych:
    val sensorEventListener = remember {
        object : SensorEventListener {
            override fun onSensorChanged(event: SensorEvent?) {
                event?.let {
                    when (it.sensor.type) {
                        Sensor.TYPE_ACCELEROMETER -> {
                            accelerometerValues = it.values.clone()
                        }

                        Sensor.TYPE_PROXIMITY -> {
                            proximityValue = it.values[0]
                        }

                    }
                }
            }

            override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int)
{

            }
        }
    }

Rejestracja nasłuchiwacza w managerze czujników dla czujników, które
istnieją z określeniem częstotliwości, z jaką system będzie wysyłać
zdarzenia zarejestrowanego czujnika:

    LaunchedEffect(Unit) {
        accelerometer?.let {
            sensorManager.registerListener(sensorEventListener, it,
SensorManager.SENSOR_DELAY_NORMAL)
        }

        proximitySensor?.let {
            sensorManager.registerListener(sensorEventListener, it,
SensorManager.SENSOR_DELAY_NORMAL)
        }

    }
// inicjalizacja firestore I auth:
    val firestore = Firebase.firestore
```

```
    val auth = FirebaseAuth.getInstance()
      Column(
            modifier = modifier,
            verticalArrangement = Arrangement.spacedBy(8.dp)
        ) {
                        Text(text = "Akcelerometr:
x=${accelerometerValues[0]} " +
                    "y=${accelerometerValues[1]} " +
                    "z=${accelerometerValues[2]}")
             Text(text = "Zbliżeniowy: $proximityValue")


            Spacer(modifier = Modifier.height(16.dp))


            Button(onClick = {
                val user = auth.currentUser
                if (user != null) {
// Mapa z danymi do zapisu wraz z czasem:
                    val sensorData = mapOf(
                        "accelerometer" to accelerometerValues.toList(),

                        "proximity" to proximityValue,

                        "timestamp" to System.currentTimeMillis()
                    )

                    // Zapis do sciezki: /users/<uid>/sensors/<id>
                    firestore
                        .collection("users")
                        .document(user.uid)
                        .collection("sensors")
                        .add(sensorData)
                        .addOnSuccessListener {
                            // Sukces
                        }
                        .addOnFailureListener {
                            // error
                        }
                }
            }) {
                Text("Zapisz do Firestore")
            }
        }
}
```

Główna aktywność, sprawdzamy, czy użytkownik jest zalogowany, jeżeli nie to wyświetlamy ekran do logowania/rejestracji, jeżeli jest zalogowany to ekran do zapisu wartości z czujników:

```
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent

import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold

import androidx.compose.runtime.Composable
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember

import androidx.compose.runtime.getValue
import androidx.compose.runtime.setValue

import androidx.compose.ui.Modifier
import com.example.firebaselogreg.ui.theme.FirebaseLogRegTheme
import com.google.firebase.auth.FirebaseAuth




class MainActivity : ComponentActivity() {
    // Inicjalizacja FirebaseAuth
    private lateinit var auth: FirebaseAuth

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        auth = FirebaseAuth.getInstance()


        setContent {

            FirebaseLogRegTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding
->

                    AppContent(
                        modifier = Modifier.padding(innerPadding),

                    )
                }
            }
        }
    }

    @Composable
    fun AppContent(modifier: Modifier) {
        //  czy user zalogowany:
        var isLoggedIn by remember { mutableStateOf(auth.currentUser !=
null) }

        if (!isLoggedIn) {
            // ekran do logowania/rejestracji:
            LogRegScreen(
                modifier=modifier,
```

```
                onLoginSuccess = {
                    isLoggedIn = true
                }
            )
        } else {
            //ekran do wysylania danych:
            MainScreen(modifier)
        }
    }
}
```

Zadanie:

Napisz aplikację umożliwiającą logowanie i rejestrację za pomocą Firebase Authentication, oraz pobranie danych z 4 czujników i zapis do firestore.

Czujniki do zapisu:

"accelerometer"

"light"

"proximity"

"magnetometer (MAGNETIC_FIELD)"