WYDZIAŁ TELEKOMUNIKACJI, INFORMATYKI I ELEKTROTECHNIKI

Programowanie aplikacji mobilnych [05-IST-PAM-SP5] - laboratorium

SPRAWOZDANIE Z LABORATORIUM

Lab Projekt, Temat sprawozdania: Projekt

Wykonali:

Gregrowicz Marcin (margre001@pbs.edu.pl)

(**05-IST-PAM-SP5**) rok 3 sem. 5 gr 4

Data: 2026-01-19

1. Cel pracy

Celem projektu jest stworzenie aplikacji mobilnej na Androida, która wykorzystuje co najmniej trzy różne sensory lub funkcje urządzenia mobilnego do akwizycji danych, ich przetwarzania oraz prezentacji użytkownikowi w atrakcyjnej formie.

Link do repozytorium Git: https://github.com/MarGregr/ProjektPogoda

Link do pobrania pliku APK: https://github.com/MarGregr/ProjektPogoda/releases/tag/APK

2. Opis aplikacji

Tematyka aplikacji

Tematyką projektu jest monitor temperatury. Aplikacja pozwala na dodawanie i przeglądanie informacji o temperaturach w określonych lokacjach. Wpisy zawierają dokładne współrzędne lokalizacji, zdjęcie, zmierzoną temperaturę oraz datę wykonanego pomiaru.

Wykorzystane sensory

W projekcie obsługiwana jest akwizycja danych z trzech źródeł:

Aparat fotograficzny – przy dodawaniu nowego wpisu można zrobić zdjęcie obecnej lokacji, ścieżka do wykonanego zdjęcia jest zapisywana we wpisie.

Lokalizacja GPS – przy dodawaniu nowego wpisu aplikacja pobiera współrzędne geograficzne pozycji urządzenia mobilnego.

Sieć/pobieranie danych z API – po pobraniu lokalizacji aplikacja korzysta z API open-meteo.com aby pobrać bieżącą temperaturę dla tej pozycji.

Reprezentacja danych

Zebrane dane są reprezentowane w formie listy pomiarów.

Ekrany

W projekcie zaimplementowałem 3 ekrany:

- Widok główny (HomeScreen)

- Widok dodawania nowego pomiaru (AddScreen)

- Widok szczegółów pomiaru (DetailsScreen)

Nawigacja pomiędzy tymi ekranami wykorzystuje Navigation Compose.

Trwałość danych

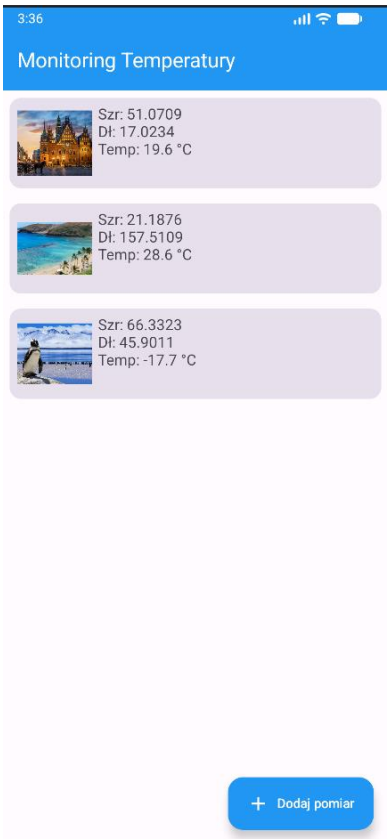
Wszystkie wpisy są zapisywane w lokalnej bazie danych SQLite z wykorzystaniem biblioteki Room.

Interakcja z użytkownikiem

Użytkownik jest w stanie wykonywać nowe pomiary, przeglądać wszystkie zapisane pomiary, a także usunąć dowolny pomiar (z poziomu widoku szczegółów).

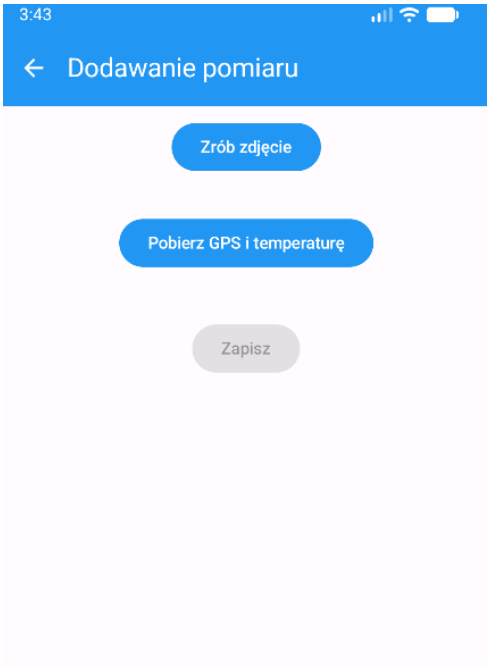
Korzystanie z aplikacji:

Widok główny:

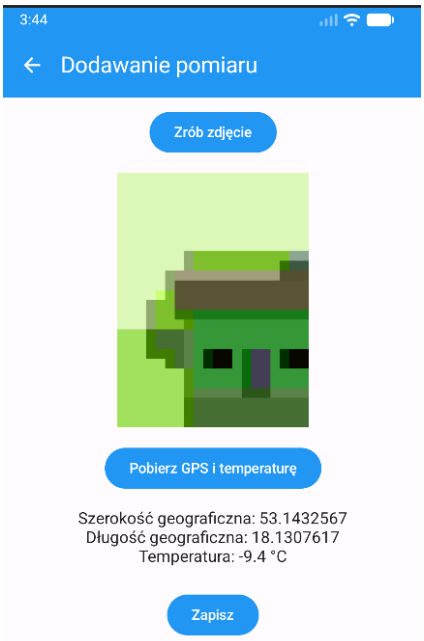


Po uruchomieniu aplikacji, użytkownikowi wyświetli się ekran główny aplikacji. Wyświetla się na nim lista wszystkich wykonanych pomiarów. Przy każdym pomiarze wyświetla się miniatura zdjęcia lokacji, jej współrzędne geograficzne oraz temperatura pobrana z API. W projekcie dodałem trzy „domyślne" pomiary, zawierające temperaturę we Wrocławiu, na Honolulu oraz na Antarktydzie wraz z odpowiednimi zdjęciami. Przy pierwszym uruchomieniu aplikacji będą widoczne więc tylko te trzy wpisy. Użytkownik może przejść do widoku ze szczegółami klikając na dowolny wpis na liście. Może także przejść do widoku dodawania nowego wpisu korzystając z pływającego przycisku „Dodaj pomiar" znajdującego się w prawym dolnym rogu ekranu.

Widok dodawania nowego pomiaru:



po zrobieniu zdjęcia i pobrania GPS oraz temperatury:



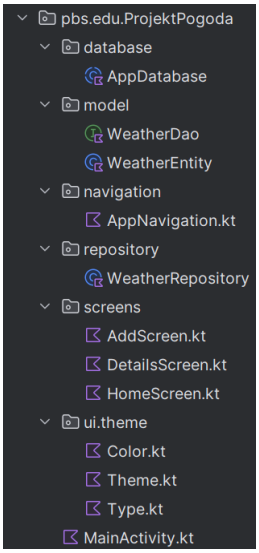
Na ekranie dodawania nowego pomiaru użytkownik może zrobić zdjęcie aparatem telefonu oraz pobrać współrzędne geograficzne i temperaturę. Program do współrzędnych korzysta z pozycji GPS telefonu, a temperaturę pobiera z wykorzystaniem API open-meteo.com. Po akwizycji wszystkich potrzebnych danych odblokowuje się przycisk „Zapisz”, który zapisuje zebrane dane do lokalnej bazy danych z wykorzystaniem biblioteki Room. Po jego wciśnięciu program wraca do widoku głównego.

Widok szczegółów pomiaru:

Po przejściu do widoku szczegółów użytkownik może zobaczyć zdjęcie, współrzędne geograficzne oraz temperaturę, a także dokładną datę wykonania tego pomiaru. Na tym ekranie użytkownik ma też opcję usunięcia pomiaru z bazy danych.

3. Kod projektu

Struktura plików w projekcie:



WeatherEntity.kt:

```kotlin
package pbs.edu.ProjektPogoda.model

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "weather_items")
data class WeatherEntity(
    @PrimaryKey(autoGenerate = true)
    val id: Long = 0,
    val imagePath: String,
    val latitude: Double,
    val longitude: Double,
    val temperature: Double,
    val timestamp: Long
)
```

WeatherDao.kt:

```kotlin
package pbs.edu.ProjektPogoda.model

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query
import kotlinx.coroutines.flow.Flow

@Dao
interface WeatherDao {
```

```
    @Query("SELECT * FROM weather_items")
    fun getAll(): Flow<List<WeatherEntity>>

    @Query("SELECT * FROM weather_items WHERE id = :id")
    suspend fun getById(id: Long): WeatherEntity?

    @Insert
    suspend fun insert(item: WeatherEntity)

    @Delete
    suspend fun deleteItem(item: WeatherEntity)
}
```

Powyższe dwa pliki odpowiadają za bazę danych. Jej pojedynczy rekord, zawierający wszystkie potrzebne atrybuty oraz metody np. dodawania lub usuwania rekordu.

WeatherRepository.kt:

```
package pbs.edu.ProjektPogoda.database

import android.content.Context
import androidx.room.Database
import androidx.room.RoomDatabase
import pbs.edu.ProjektPogoda.model.WeatherDao
import pbs.edu.ProjektPogoda.model.WeatherEntity
import androidx.room.Room
import androidx.sqlite.db.SupportSQLiteDatabase
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.launch


@Database(entities = [WeatherEntity::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun weatherDao(): WeatherDao

    companion object {
        @Volatile private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context, scope: CoroutineScope): AppDatabase =
            INSTANCE ?: synchronized(this) {
                INSTANCE ?: Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "weather_db"
                ).fallbackToDestructiveMigration()
                    .addCallback(AppDatabaseCallback(scope))
                    .build().also { INSTANCE = it }
            }
    }

    private class AppDatabaseCallback(private val scope: CoroutineScope) :
RoomDatabase.Callback() {
        override fun onCreate(db: SupportSQLiteDatabase) {
            super.onCreate(db)
            INSTANCE?.let { database ->
                scope.launch {
                    val dao = database.weatherDao()

                    val sample = WeatherEntity(
                        imagePath =
"android.resource://pbs.edu.ProjektPogoda/drawable/wroclaw",
                        latitude = 51.0709,
                        longitude = 17.0234,
                        temperature = 19.6,
                        timestamp = 1749725686000
                    )

                    val sample2 = WeatherEntity(
                        imagePath =
"android.resource://pbs.edu.ProjektPogoda/drawable/honolulu",
                        latitude = 21.1876,
                        longitude = 157.5109,
```

```
                        temperature = 28.6,
                        timestamp = 1749725686000
                    )

                    val sample3 = WeatherEntity(
                        imagePath =
"android.resource://pbs.edu.ProjektPogoda/drawable/antarktyda",
                        latitude = 66.3323,
                        longitude = 45.9011,
                        temperature = -17.7,
                        timestamp = 1749725686000
                    )
                    dao.insert(sample)
                    dao.insert(sample2)
                    dao.insert(sample3)
                }
            }
        }
    }
}
```

Plik odpowiada za bazę danych a także wpisanie do niej domyślnych wpisów

WeatherRepository.kt:

```
package pbs.edu.ProjektPogoda.repository

import android.content.Context
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.flow.Flow
import pbs.edu.ProjektPogoda.database.AppDatabase
import pbs.edu.ProjektPogoda.model.WeatherEntity

class WeatherRepository(context: Context) {

    private val dao = AppDatabase.getDatabase(context,
CoroutineScope(Dispatchers.IO)).weatherDao()

    val items: Flow<List<WeatherEntity>> = dao.getAll()

    suspend fun addItem(item: WeatherEntity) {
        dao.insert(item)
    }

    suspend fun deleteItem(item: WeatherEntity) {
        dao.deleteItem(item)
    }

    suspend fun getItem(id: Long): WeatherEntity? {
        return dao.getById(id)
    }
}
```

Color.kt:

```
package pbs.edu.ProjektPogoda.ui.theme

import androidx.compose.ui.graphics.Color
val Purple200 = Color(0xFFBB86FC)
val Purple500 = Color(0xFF6200EE)
val Purple700 = Color(0xFF3700B3)
val Teal200 = Color(0xFF03DAC5)
val AppBlue = Color(0xFF2196F3)
val White = Color(0xFFFFFFFF)
```

Type.kt:

```
package pbs.edu.ProjektPogoda.ui.theme

import androidx.compose.material3.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

val Typography = Typography(
    bodyLarge = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
)
```

Theme.kt:

```
package pbs.edu.ProjektPogoda.ui.theme

import android.app.Activity
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.runtime.SideEffect
import androidx.compose.ui.graphics.toArgb
import androidx.compose.ui.platform.LocalView

private val DarkColorScheme = darkColorScheme(
    primary = AppBlue,
    secondary = Teal200,
    tertiary = Purple700
)
private val LightColorScheme = lightColorScheme(
    primary = AppBlue,
    secondary = White,
    tertiary = Purple700
)

@Composable
fun ProjektPogodaTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () -> Unit
) {
    val colorScheme = if (darkTheme) DarkColorScheme else LightColorScheme

    val view = LocalView.current
    if (!view.isInEditMode) {
        SideEffect {
            val window = (view.context as Activity).window
            window.statusBarColor = colorScheme.primary.toArgb()
        }
    }

    MaterialTheme(
        colorScheme = if (darkTheme) DarkColorScheme else LightColorScheme,
        typography = Typography,
        content = content
    )
}
```

Powyższe trzy pliki obsługują wizualny wygląd aplikacji.

HomeScreen.kt:

```
package pbs.edu.ProjektPogoda.screens

import androidx.compose.foundation.Image
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Add
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import coil.compose.rememberAsyncImagePainter
import pbs.edu.ProjektPogoda.repository.WeatherRepository
import pbs.edu.ProjektPogoda.ui.theme.AppBlue
import pbs.edu.ProjektPogoda.ui.theme.White

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun HomeScreen(navController: NavController) {
    val context = LocalContext.current
    val repo = remember { WeatherRepository(context) }
    val items by repo.items.collectAsState(initial = emptyList())

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(text = "Monitoring Temperatury") },
                colors = TopAppBarDefaults.topAppBarColors(containerColor = AppBlue,
titleContentColor = White)
            )
        },
        floatingActionButton = {
            ExtendedFloatingActionButton(
                onClick = { navController.navigate("add") },
                containerColor = AppBlue,
                contentColor = White
            ) {
                Icon(Icons.Default.Add, contentDescription = null)
                Spacer(Modifier.width(8.dp))
                Text(text = "Dodaj pomiar")
            }
        }
    ) { padding ->
        LazyColumn(modifier = Modifier.padding(padding)) {
            items(items) { item ->
                Card(
                    modifier = Modifier
                        .fillMaxWidth()
                        .padding(8.dp),
                    onClick = {
                        navController.navigate("details/${item.id}")
                    }
                ) {
                    Row(modifier = Modifier.padding(8.dp)) {
                        Image(
                            painter = rememberAsyncImagePainter(item.imagePath),
                            contentDescription = null,
                            modifier = Modifier.size(80.dp)
                        )
                        Spacer(modifier = Modifier.width(6.dp))
                        Column {
                            Text("Szr: ${item.latitude}")
                            Text("Dł: ${item.longitude}")
```

```
                            Text("Temp: ${item.temperature} °C")
                        }
                    }

                }
            }
        }
    }
}
```

AddScreen.kt:

```
package pbs.edu.ProjektPogoda.screens

import android.Manifest
import android.content.pm.PackageManager
import android.location.Location
import android.net.Uri
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import androidx.core.content.ContextCompat
import androidx.core.content.FileProvider
import androidx.navigation.NavController
import coil.compose.rememberAsyncImagePainter
import com.google.android.gms.location.LocationServices
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import pbs.edu.ProjektPogoda.model.WeatherEntity
import pbs.edu.ProjektPogoda.repository.WeatherRepository
import pbs.edu.ProjektPogoda.ui.theme.AppBlue
import pbs.edu.ProjektPogoda.ui.theme.White
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import retrofit2.http.GET
import retrofit2.http.Query
import java.io.File

data class WeatherResponse(
    val current_weather: CurrentWeather
)

data class CurrentWeather(
    val temperature: Double
)

interface WeatherApi {
    @GET("v1/forecast")
    suspend fun getWeather(
        @Query("latitude") lat: Double,
        @Query("longitude") lon: Double,
        @Query("current_weather") current: Boolean = true
    ): WeatherResponse
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun AddScreen(navController: NavController) {

    val scrollState = rememberScrollState()
    val context = LocalContext.current
    val repo = remember { WeatherRepository(context) }
    val fusedLocationClient = remember {
```

```
        LocationServices.getFusedLocationProviderClient(context)
    }

    var imageUri by remember { mutableStateOf<Uri?>(null) }
    var tempUri by remember { mutableStateOf<Uri?>(null) }
    var latitude by remember { mutableStateOf<Double?>(null) }
    var longitude by remember { mutableStateOf<Double?>(null) }
    var temperature by remember { mutableStateOf<Double?>(null) }

    val api = remember {
        Retrofit.Builder()
            .baseUrl("https://api.open-meteo.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(WeatherApi::class.java)
    }

    fun createImageUri(): Uri {
        val file = File(context.cacheDir, "photo_${System.currentTimeMillis()}.jpg")
        return FileProvider.getUriForFile(
            context,
            "${context.packageName}.provider",
            file
        )
    }


    val cameraLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.TakePicture()
    ) { success ->
        if (!success) {
            imageUri = null
        } else {
            imageUri = tempUri
        }
    }

    val cameraPermissionLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.RequestMultiplePermissions()
    ) { permissions ->
        val cameraGranted = permissions[Manifest.permission.CAMERA] ?: false

        if (cameraGranted) {
            val uri = createImageUri()
            tempUri = uri
            cameraLauncher.launch(uri)
        }
    }

    fun fetchLocationAndWeather() {
        fusedLocationClient.lastLocation.addOnSuccessListener { location: Location? ->
            location?.let {
                latitude = it.latitude
                longitude = it.longitude

                CoroutineScope(Dispatchers.IO).launch {
                    try {
                        val response = api.getWeather(latitude!!, longitude!!)
                        temperature = response.current_weather.temperature
                    } catch (e: Exception) {
                        e.printStackTrace()
                    }
                }
            }
        }
    }

    val locationPermissionLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.RequestMultiplePermissions()
    ) { permissions ->
        val locationGranted = permissions[Manifest.permission.ACCESS_FINE_LOCATION] ?: false

        if (locationGranted) {
            fetchLocationAndWeather()
        }
    }
```

```
    fun checkLocationPermission() {
        if (
            ContextCompat.checkSelfPermission(context,
Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED
        ) {
            locationPermissionLauncher.launch(
                arrayOf(
                    Manifest.permission.ACCESS_FINE_LOCATION
                )
            )
            return
        }
        fetchLocationAndWeather()
    }

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(text = "Dodawanie pomiaru") },
                colors = TopAppBarDefaults.topAppBarColors(containerColor = AppBlue,
titleContentColor = White),
                navigationIcon = {
                    IconButton(onClick = { navController.popBackStack() }) {
                        Icon(
                            imageVector = Icons.Default.ArrowBack,
                            contentDescription = "Wróć",
                            tint = White
                        )
                    }
                }
            )
        }
    ) { paddingValues ->
        Column(
            Modifier
                .padding(paddingValues)
                .verticalScroll(scrollState)
                .fillMaxWidth(),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {

            Spacer(modifier = Modifier.height(10.dp))

            Button(onClick = {
                if (ContextCompat.checkSelfPermission(context, Manifest.permission.CAMERA)
                    == PackageManager.PERMISSION_GRANTED
                ) {

                    val uri = createImageUri()
                    tempUri = uri
                    cameraLauncher.launch(uri)

                } else {
                    cameraPermissionLauncher.launch(arrayOf(Manifest.permission.CAMERA))
                }
            },
                colors = ButtonDefaults.buttonColors(
                    containerColor = AppBlue,
                    contentColor = White
                )
            ) {
                Text("Zrób zdjęcie")
            }

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

            imageUri?.let {
                Image(
                    painter = rememberAsyncImagePainter(it),
                    contentDescription = "Photo",
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(250.dp)
                )
            }
```

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

            Button(
                onClick = { checkLocationPermission() },
                colors = ButtonDefaults.buttonColors(
                    containerColor = AppBlue,
                    contentColor = androidx.compose.ui.graphics.Color.White
                )
            ) {
                Text(text = "Pobierz GPS i temperaturę")
            }

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

            latitude?.let { Text("Szerokość geograficzna: $it") }
            longitude?.let { Text("Długość geograficzna: $it") }
            temperature?.let { Text("Temperatura: $it °C") }

            Spacer(modifier = Modifier.height(24.dp))

            Button(
                enabled = imageUri != null && latitude != null && temperature != null,
                onClick = {
                    CoroutineScope(Dispatchers.IO).launch {
                        repo.addItem(
                            WeatherEntity(
                                imagePath = imageUri.toString(),
                                latitude = latitude!!,
                                longitude = longitude!!,
                                temperature = temperature!!,
                                timestamp = System.currentTimeMillis()
                            )
                        )
                    }
                    navController.popBackStack()
                },
                colors = ButtonDefaults.buttonColors(
                    containerColor = AppBlue,
                    contentColor = White
                )
            ) {
                Text("Zapisz")
            }
        }
    }
}
```

DetailsScreen.kt:

```
package pbs.edu.ProjektPogoda.screens

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import coil.compose.rememberAsyncImagePainter
import kotlinx.coroutines.launch
import pbs.edu.ProjektPogoda.model.WeatherEntity
import pbs.edu.ProjektPogoda.repository.WeatherRepository
import pbs.edu.ProjektPogoda.ui.theme.AppBlue
import pbs.edu.ProjektPogoda.ui.theme.White
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DetailsScreen(
    navController: NavController,
    itemId: Long
) {
    val scrollState = rememberScrollState()
    val context = LocalContext.current
    val repo = remember { WeatherRepository(context) }
    val scope = rememberCoroutineScope()
    var item by remember { mutableStateOf<WeatherEntity?>(null) }

    LaunchedEffect(itemId) {
        item = repo.getItem(itemId)
    }

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(text = "Szczegóły pomiaru") },
                colors = TopAppBarDefaults.topAppBarColors(containerColor = AppBlue,
titleContentColor = White),


                navigationIcon = {
                    IconButton(onClick = { navController.popBackStack() }) {
                        Icon(
                            imageVector = Icons.Default.ArrowBack,
                            contentDescription = "Wróć",
                            tint = White
                        )
                    }
                }
            )
        }
    ) { paddingValues ->

        Column(modifier = Modifier.padding(paddingValues)) {
            Surface(
                modifier = Modifier
                    .fillMaxHeight()
                    .fillMaxWidth()
                    .verticalScroll(scrollState)
            ) {
                Column(
                    horizontalAlignment = Alignment.CenterHorizontally,
                    verticalArrangement = Arrangement.Center
                ) {

                    item?.let { weather ->

                        Image(
                            painter = rememberAsyncImagePainter(weather.imagePath),
                            contentDescription = "Photo",
                            modifier = Modifier
                                .fillMaxWidth()
                                .height(250.dp)
                        )

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

                        Text(
                            text = "Szerokość geograficzna: ${weather.latitude}",
                            style = MaterialTheme.typography.headlineSmall
                        )

                        Text(
                            text = "Długość geograficzna: ${weather.longitude}",
                            style = MaterialTheme.typography.headlineSmall
                        )

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

                        Text(
                            text = "Temperatura: ${weather.temperature} °C",
                            style = MaterialTheme.typography.headlineMedium
```

```
                        )

                        Spacer(modifier = Modifier.height(40.dp))

                        val sdf = remember { SimpleDateFormat(
                                "yyyy-MM-dd HH:mm",
                                Locale.getDefault()
                            )
                        }
                        val dateString = sdf.format(Date(weather.timestamp))

                        Text(
                            text = "Data odczytu: ${dateString}",
                            style = MaterialTheme.typography.headlineMedium
                        )

                        Spacer(modifier = Modifier.height(40.dp))

                        Button(
                            onClick = {
                                scope.launch {
                                    item?.let {
                                        repo.deleteItem(it)
                                        navController.popBackStack()
                                    }
                                }
                            },
                            colors = ButtonDefaults.buttonColors(containerColor = Color.Red)
                        ) {
                            Text(text = "Usuń pomiar", color = White)
                        }

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

                        Button(
                            onClick = { navController.popBackStack() },
                            colors = ButtonDefaults.buttonColors(
                                containerColor = AppBlue,
                                contentColor = Color.White
                            )
                        ) {
                            Text(text = "Wróć")
                        }
                    } ?: run {
                        CircularProgressIndicator()
                    }
                }
            }
        }
    }
}
```

Powyższe trzy pliki są odpowiedzialne za trzy ekrany projektu, widok główny, widok dodawania nowego pomiaru oraz widok szczegółów pomiaru.

AppNavigation.kt:

```
package edu.pbs.ProjektPogoda.navigation

import androidx.compose.runtime.Composable
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import androidx.navigation.NavType
import androidx.navigation.navArgument
import pbs.edu.ProjektPogoda.screens.AddScreen
import pbs.edu.ProjektPogoda.screens.DetailsScreen
import pbs.edu.ProjektPogoda.screens.HomeScreen

@Composable
fun AppNavigation() {
```

```
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = "home"
    ) {

        composable("home") {
            HomeScreen(navController = navController)
        }

        composable("add") {
            AddScreen(navController = navController)
        }

        composable(
            route = "details/{id}",
            arguments = listOf(
                navArgument("id") {
                    type = NavType.LongType
                }
            )
        ) { backStackEntry ->

            val id = backStackEntry.arguments?.getLong("id") ?: 0L

            DetailsScreen(
                navController = navController,
                itemId = id
            )
        }
    }
}
```

Plik obsługuje nawigację pomiędzy różnymi widokami w projekcie.

MainActivity.kt:

```
package pbs.edu.ProjektPogoda

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.runtime.Composable
import androidx.compose.ui.tooling.preview.Preview
import edu.pbs.ProjektPogoda.navigation.AppNavigation
import pbs.edu.ProjektPogoda.ui.theme.ProjektPogodaTheme

class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ProjektPogodaApp()
        }
    }
}

@Composable
fun ProjektPogodaApp() {
    ProjektPogodaTheme {
        AppNavigation()
    }
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    ProjektPogodaApp()
}
```

Plik uruchamiający aplikację.

Wnioski:

Praktyczne zastosowanie funkcji navController.popBackStack() w połączeniu z ikonami strzałek w top barze aplikacji oraz dedykowanymi przyciskami powrotu pokazało, jak efektywnie zarządzać stosem ekranów, zapewniając spójność nawigacyjną i wygodę obsługi aplikacji.

Wykorzystanie biblioteki Room do obsługi bazy danych SQLite pozwoliło na trwałe przechowywanie zebranych informacji o temperaturze, lokalizacji i zdjęciach, co jest kluczowe dla funkcjonalności aplikacji typu monitor danych.

Integracja zewnętrznych interfejsów API (open-meteo.com) wraz z sensorami urządzenia (GPS oraz aparat) umożliwiła stworzenie kompleksowego narzędzia do akwizycji danych środowiskowych w czasie rzeczywistym.