Unidad Didáctica 6: EJEMPLO 6

Programación Dirigida por Eventos

1. Enunciado

Enunciado

En este ejercicio práctico, desarrollaremos una aplicación Android que integre funciones geográficas y de mapas, siguiendo los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030, específicamente el ODS 13: Acción por el Clima. La aplicación ayudará a los usuarios a identificar y reportar áreas afectadas por el cambio climático.

Ejercicio 6: Aplicación de Reporte de Áreas Afectadas por el Cambio Climático

---

Introducción

El cambio climático tiene un impacto significativo en diversas áreas geográficas. Con la ayuda de aplicaciones móviles que integren funciones geográficas, los usuarios pueden identificar y reportar áreas afectadas, contribuyendo a una mejor comprensión y acción sobre el clima. Este ejercicio se centra en desarrollar una aplicación que permita a los usuarios utilizar mapas y geolocalización para reportar dichas áreas.

Enunciado del Problema

Desarrollar una aplicación Android que permita a los usuarios:

1. Ver un mapa interactivo de su ubicación actual.
2. Marcar y reportar áreas afectadas por el cambio climático.
3. Obtener información sobre las áreas reportadas.
4. Visualizar estadísticas y gráficos relacionados con los reportes.

2. Solución

A continuación, se presenta una solución detallada para desarrollar la aplicación propuesta.

Paso 1: Configuración del Proyecto

1. Iniciar un nuevo proyecto en Android Studio con una "Actividad Vacía".
2. Configurar los archivos build.gradle para asegurarse de tener las dependencias necesarias.

gradle

```
Copiar código
// build.gradle (Project level)
allprojects {
    repositories {
        google()
        mavenCentral()
    }
}

// build.gradle (Module level)
dependencies {
    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'
    implementation 'com.google.android.gms:play-services-maps:17.0.1'
    implementation 'com.google.android.gms:play-services-location:18.0.0'
}
```

Paso 2: Diseño de la Interfaz de Usuario

Crear un archivo XML para la actividad principal (activity_main.xml) que incluya un MapView para mostrar el mapa, un Button para marcar áreas afectadas y un RecyclerView para mostrar la lista de reportes.

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.gms.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/buttonMarkArea"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

    <Button
        android:id="@+id/buttonMarkArea"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Marcar Área"
        app:layout_constraintBottom_toTopOf="@id/recyclerView"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>
```

```
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toBottomOf="@id/buttonMarkArea"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Paso 3: Implementación de la Actividad Principal

Implementar la lógica en MainActivity.java para manejar el mapa y la funcionalidad de marcar áreas afectadas.

```
package com.example.climatechangeapp;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.location.Location;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnSuccessListener;

import java.util.List;

public class MainActivity extends AppCompatActivity implements OnMapReadyCallback {

    private MapView mapView;
    private Button buttonMarkArea;
    private RecyclerView recyclerView;
    private FusedLocationProviderClient fusedLocationClient;
    private GoogleMap googleMap;
```

```
    private ReportViewModel reportViewModel;
    private ReportAdapter reportAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mapView = findViewById(R.id.mapView);
        buttonMarkArea = findViewById(R.id.buttonMarkArea);
        recyclerView = findViewById(R.id.recyclerView);

        mapView.onCreate(savedInstanceState);
        mapView.getMapAsync(this);

        fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setHasFixedSize(true);
        reportAdapter = new ReportAdapter();
        recyclerView.setAdapter(reportAdapter);

        reportViewModel = new ViewModelProvider(this).get(ReportViewModel.class);

        reportViewModel.getAllReports().observe(this, new Observer<List<Report>>() {
            @Override
            public void onChanged(List<Report> reports) {
                reportAdapter.setReports(reports);
            }
        });

        buttonMarkArea.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                markArea();
            }
        });
    }

    private void markArea() {
        fusedLocationClient.getLastLocation()
            .addOnSuccessListener(this, new OnSuccessListener<Location>() {
                @Override
                public void onSuccess(Location location) {
                    if (location != null) {
                        LatLng currentLatLng = new LatLng(location.getLatitude(), location.getLongitude());
                        googleMap.addMarker(new MarkerOptions().position(currentLatLng).title("Área
Afectada"));
                        reportViewModel.insert(new Report(currentLatLng.latitude, currentLatLng.longitude,
"Área Afectada"));
                    } else {
```

```
                Toast.makeText(MainActivity.this, "No se pudo obtener la ubicación",
Toast.LENGTH_SHORT).show();
            }
          }
        });
    }

    @Override
    public void onMapReady(GoogleMap map) {
        googleMap = map;
        googleMap.getUiSettings().setZoomControlsEnabled(true);

        fusedLocationClient.getLastLocation()
            .addOnSuccessListener(this, new OnSuccessListener<Location>() {
                @Override
                public void onSuccess(Location location) {
                    if (location != null) {
                        LatLng currentLatLng = new LatLng(location.getLatitude(), location.getLongitude());
                        googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(currentLatLng, 15));
                    }
                }
            });
    }

    @Override
    protected void onResume() {
        super.onResume();
        mapView.onResume();
    }

    @Override
    protected void onPause() {
        super.onPause();
        mapView.onPause();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        mapView.onDestroy();
    }

    @Override
    public void onLowMemory() {
        super.onLowMemory();
        mapView.onLowMemory();
    }
}
```

Paso 4: Configuración de Room para la Base de Datos

Configurar Room para manejar la base de datos local de los reportes.

Report.java

```
package com.example.climatechangeapp;

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

@Entity(tableName = "report_table")
public class Report {
    @PrimaryKey(autoGenerate = true)
    private int id;

    private double latitude;
    private double longitude;
    private String description;

    public Report(double latitude, double longitude, String description) {
        this.latitude = latitude;
        this.longitude = longitude;
        this.description = description;
    }

    public int getId() { return id; }
    public double getLatitude() { return latitude; }
    public double getLongitude() { return longitude; }
    public String getDescription() { return description; }

    public void setId(int id) { this.id = id; }
}
```

ReportDao.java

```
package com.example.climatechangeapp;

import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Insert;
import androidx.room.Query;

import java.util.List;

@Dao
public interface ReportDao {
    @Insert
    void insert(Report report);

    @Query("DELETE FROM report_table")
    void deleteAll();

    @Query("SELECT * FROM report_table ORDER BY id DESC")
    LiveData<List<Report>> getAllReports();
}
```

ReportDatabase.java

```
package com.example.climatechangeapp;

import android.content.Context;

import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;

@Database(entities = {Report.class}, version = 1)
public abstract class ReportDatabase extends RoomDatabase {
   private static ReportDatabase instance;

   public abstract ReportDao reportDao();

   public static synchronized ReportDatabase getInstance(Context context) {
      if (instance == null) {
         instance = Room.databaseBuilder(context.getApplicationContext(),
               ReportDatabase.class, "report_database")
               .fallbackToDestructiveMigration()
               .build();
      }
      return instance;
   }
}
```

ReportRepository.java

```
package com.example.climatechangeapp;

import android.app.Application;
import androidx.lifecycle.LiveData;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ReportRepository {
   private ReportDao reportDao;
   private LiveData<List<Report>> allReports;
   private final ExecutorService executorService;

   public ReportRepository(Application application) {
      ReportDatabase database = ReportDatabase.getInstance(application);
      reportDao = database.reportDao();
      allReports = reportDao.getAllReports();
      executorService = Executors.newFixedThreadPool(2);
   }

   public void insert(Report report) {
      executorService.execute(() -> reportDao.insert(report));
   }
```

```
    public LiveData<List<Report>> getAllReports() {
        return allReports;
    }
}
```

Paso 5: Implementación del ViewModel

Crear una clase ReportViewModel.java para manejar los datos de los reportes y proporcionar métodos para interactuar con el repositorio.

```
package com.example.climatechangeapp;

import android.app.Application;
import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;
import java.util.List;

public class ReportViewModel extends AndroidViewModel {
    private ReportRepository repository;
    private LiveData<List<Report>> allReports;

    public ReportViewModel(@NonNull Application application) {
        super(application);
        repository = new ReportRepository(application);
        allReports = repository.getAllReports();
    }

    public LiveData<List<Report>> getAllReports() {
        return allReports;
    }

    public void insert(Report report) {
        repository.insert(report);
    }
}
```

Paso 6: Implementación del Adaptador para RecyclerView

Crear una clase ReportAdapter.java para mostrar los reportes en un RecyclerView.

```
package com.example.climatechangeapp;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;
import java.util.List;
```

```
public class ReportAdapter extends RecyclerView.Adapter<ReportAdapter.ReportHolder> {
   private List<Report> reports = new ArrayList<>();

   @NonNull
   @Override
   public ReportHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
      View itemView = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.report_item, parent, false);
      return new ReportHolder(itemView);
   }

   @Override
   public void onBindViewHolder(@NonNull ReportHolder holder, int position) {
      Report currentReport = reports.get(position);
      holder.textViewLatitude.setText(String.valueOf(currentReport.getLatitude()));
      holder.textViewLongitude.setText(String.valueOf(currentReport.getLongitude()));
      holder.textViewDescription.setText(currentReport.getDescription());
   }

   @Override
   public int getItemCount() {
      return reports.size();
   }

   public void setReports(List<Report> reports) {
      this.reports = reports;
      notifyDataSetChanged();
   }

   class ReportHolder extends RecyclerView.ViewHolder {
      private TextView textViewLatitude;
      private TextView textViewLongitude;
      private TextView textViewDescription;

      public ReportHolder(View itemView) {
         super(itemView);
         textViewLatitude = itemView.findViewById(R.id.textViewLatitude);
         textViewLongitude = itemView.findViewById(R.id.textViewLongitude);
         textViewDescription = itemView.findViewById(R.id.textViewDescription);
      }
   }
}
```

report_item.xml

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:padding="8dp">
```

```
    <TextView
        android:id="@+id/textViewLatitude"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Latitud"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@id/textViewLongitude"
        app:layout_constraintHorizontalChainStyle="packed"/>

    <TextView
        android:id="@+id/textViewLongitude"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Longitud"
        app:layout_constraintStart_toEndOf="@id/textViewLatitude"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@id/textViewDescription"/>

    <TextView
        android:id="@+id/textViewDescription"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Descripción"
        app:layout_constraintStart_toEndOf="@id/textViewLongitude"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Conclusión

Este ejercicio práctico permite a los estudiantes aplicar conceptos de desarrollo de aplicaciones Android para crear una herramienta útil que contribuye al ODS 13: Acción por el Clima. A través de esta actividad, los estudiantes desarrollan habilidades en el uso de mapas y geolocalización, la gestión de datos y la implementación de interfaces de usuario interactivas.

UAX Universidad Alfonso X el Sabio

G R A C I A S

UAX.COM