Unidad Didáctica 5: EJEMPLO 5

Programación Dirigida por Eventos

1. Enunciado

En este ejercicio práctico, desarrollaremos una aplicación Android optimizada para ser rápida y liviana, siguiendo los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030, específicamente el ODS 11: Ciudades y Comunidades Sostenibles. La aplicación ayudará a los usuarios a gestionar el transporte público y otras formas de movilidad sostenible de manera eficiente.

Ejercicio 5: Aplicación de Transporte Público Eficiente

Introducción

La eficiencia en el transporte público es clave para la sostenibilidad urbana y la reducción de la huella de carbono. Las aplicaciones móviles pueden optimizar el uso del transporte público proporcionando horarios en tiempo real, rutas optimizadas y alertas de servicio. Este ejercicio se centra en desarrollar una aplicación que sea rápida y liviana, optimizando el rendimiento y la experiencia del usuario.

Enunciado del Problema

Desarrollar una aplicación Android que permita a los usuarios:

1. Consultar horarios de transporte público en tiempo real.
2. Obtener rutas optimizadas basadas en la ubicación del usuario.
3. Recibir alertas de servicio y notificaciones.
4. Minimizar el consumo de datos y la carga en la batería del dispositivo.

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.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'com.github.bumptech.glide:glide:4.12.0'
}
```

Paso 2: Diseño de la Interfaz de Usuario

Crear un archivo XML para la actividad principal (activity_main.xml) que incluya un RecyclerView para mostrar los horarios de transporte público, un Button para obtener rutas y un TextView para mostrar alertas de servicio.

```
<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">

    <RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/buttonGetRoutes"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

    <Button
        android:id="@+id/buttonGetRoutes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Obtener Rutas"
        app:layout_constraintBottom_toTopOf="@id/textViewAlerts"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>
```

```
    <TextView
        android:id="@+id/textViewAlerts"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Alertas de Servicio"
        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 la consulta de horarios, la obtención de rutas y la recepción de alertas de servicio.

```
package com.example.publictransport;

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.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

public class MainActivity extends AppCompatActivity {

    private RecyclerView recyclerView;
    private Button buttonGetRoutes;
    private TextView textViewAlerts;
    private TransportViewModel transportViewModel;
    private TransportAdapter transportAdapter;

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

        recyclerView = findViewById(R.id.recyclerView);
        buttonGetRoutes = findViewById(R.id.buttonGetRoutes);
        textViewAlerts = findViewById(R.id.textViewAlerts);

        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setHasFixedSize(true);
```

```
        transportAdapter = new TransportAdapter();
        recyclerView.setAdapter(transportAdapter);

        transportViewModel = new ViewModelProvider(this).get(TransportViewModel.class);

        transportViewModel.getTransportData().observe(this, new Observer<List<Transport>>() {
            @Override
            public void onChanged(List<Transport> transportData) {
                transportAdapter.setTransportData(transportData);
            }
        });

        transportViewModel.getAlerts().observe(this, new Observer<String>() {
            @Override
            public void onChanged(String alert) {
                textViewAlerts.setText(alert);
            }
        });

        buttonGetRoutes.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                transportViewModel.fetchRoutes();
            }
        });
    }
}
```

Paso 4: Configuración de Retrofit para la API

Configurar Retrofit para manejar las solicitudes de la API de transporte público.

ApiService.java

java
Copiar código

```
package com.example.publictransport;

import java.util.List;

import retrofit2.Call;
import retrofit2.http.GET;

public interface ApiService {
    @GET("getTransportData")
    Call<List<Transport>> getTransportData();

    @GET("getAlerts")
    Call<String> getAlerts();
}
```

RetrofitClient.java

java
Copiar código

```
package com.example.publictransport;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitClient {
    private static Retrofit retrofit;
    private static final String BASE_URL = "https://api.publictransport.com/";

    public static Retrofit getRetrofitInstance() {
        if (retrofit == null) {
            retrofit = new retrofit2.Retrofit.Builder()
                    .baseUrl(BASE_URL)
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();
        }
        return retrofit;
    }
}
```

Paso 5: Implementación del ViewModel

Crear una clase TransportViewModel.java para manejar los datos de transporte y proporcionar métodos para interactuar con la API.

```
package com.example.publictransport;

import android.app.Application;

import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class TransportViewModel extends AndroidViewModel {
    private MutableLiveData<List<Transport>> transportData;
    private MutableLiveData<String> alerts;

    public TransportViewModel(@NonNull Application application) {
        super(application);
        transportData = new MutableLiveData<>();
        alerts = new MutableLiveData<>();
    }
```

```
    public LiveData<List<Transport>> getTransportData() {
        return transportData;
    }

    public LiveData<String> getAlerts() {
        return alerts;
    }

    public void fetchRoutes() {
        ApiService apiService = RetrofitClient.getRetrofitInstance().create(ApiService.class);
        Call<List<Transport>> call = apiService.getTransportData();
        call.enqueue(new Callback<List<Transport>>() {
            @Override
            public void onResponse(Call<List<Transport>> call, Response<List<Transport>> response) {
                transportData.setValue(response.body());
            }

            @Override
            public void onFailure(Call<List<Transport>> call, Throwable t) {
                // Manejar errores
            }
        });

        Call<String> alertsCall = apiService.getAlerts();
        alertsCall.enqueue(new Callback<String>() {
            @Override
            public void onResponse(Call<String> call, Response<String> response) {
                alerts.setValue(response.body());
            }

            @Override
            public void onFailure(Call<String> call, Throwable t) {
                // Manejar errores
            }
        });
    }
}
```

Paso 6: Implementación del Adaptador para RecyclerView

Crear una clase TransportAdapter.java para mostrar los datos de transporte en un RecyclerView.

```
package com.example.publictransport;

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 TransportAdapter extends RecyclerView.Adapter<TransportAdapter.TransportHolder> {
   private List<Transport> transportData = new ArrayList<>();

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

   @Override
   public void onBindViewHolder(@NonNull TransportHolder holder, int position) {
      Transport currentTransport = transportData.get(position);
      holder.textViewLine.setText(currentTransport.getLine());
      holder.textViewTime.setText(currentTransport.getTime());
   }

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

   public void setTransportData(List<Transport> transportData) {
      this.transportData = transportData;
      notifyDataSetChanged();
   }

   class TransportHolder extends RecyclerView.ViewHolder {
      private TextView textViewLine;
      private TextView textViewTime;

      public TransportHolder(View itemView) {
         super(itemView);
         textViewLine = itemView.findViewById(R.id.textViewLine);
         textViewTime = itemView.findViewById(R.id.textViewTime);
      }
   }
}
```

transport_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/textViewLine"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Línea"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@id/textViewTime"
        app:layout_constraintHorizontalChainStyle="packed"/>

    <TextView
        android:id="@+id/textViewTime"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Hora"
        app:layout_constraintStart_toEndOf="@id/textViewLine"
        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 11: Ciudades y Comunidades Sostenibles. A través de esta actividad, los estudiantes desarrollan habilidades en la optimización del rendimiento de aplicaciones móviles, la gestión eficiente de datos y la mejora de la experiencia del usuario.

UAX Universidad Alfonso X el Sabio

G R A C I A S

UAX.COM