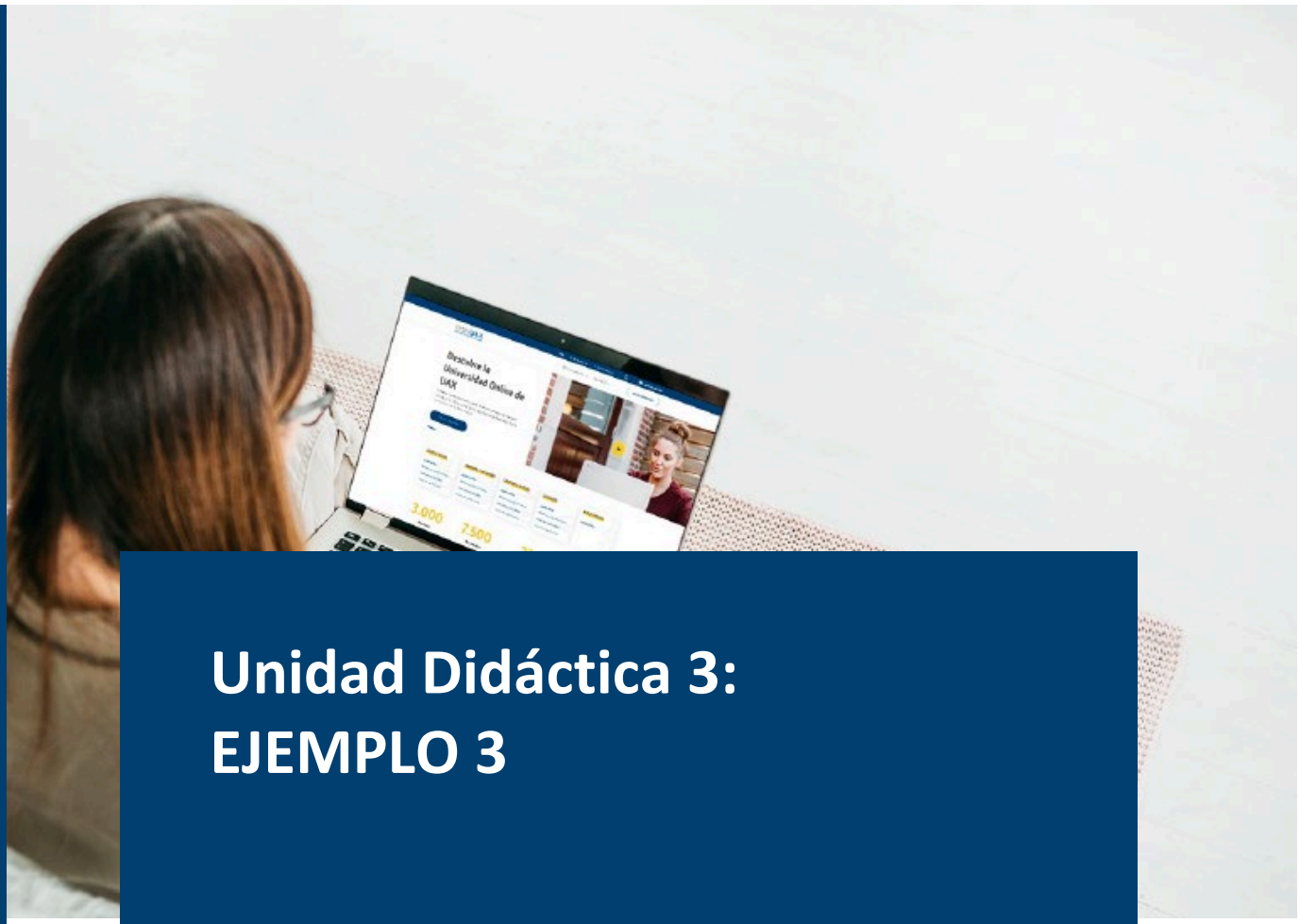

Unidad Didáctica 3: EJEMPLO 3

Programación Dirigida a Eventos

1. Enunciado

En este ejercicio práctico, desarrollaremos una aplicación Android que permita guardar datos del usuario de manera eficiente y segura, siguiendo los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030, específicamente el ODS 3: Salud y Bienestar. La aplicación ayudará a los usuarios a llevar un registro de su historial médico personal.

Ejercicio 3: Historial Médico Electrónico Personal

Introducción

La gestión del historial médico personal es esencial para asegurar un seguimiento adecuado de la salud y facilitar la comunicación con profesionales médicos. Con la ayuda de aplicaciones móviles, los usuarios pueden almacenar, actualizar y acceder a su información médica de manera segura. Este ejercicio se centra en desarrollar una aplicación que permita a los usuarios guardar y gestionar su historial médico personal.

Enunciado del Problema

Desarrollar una aplicación Android que permita a los usuarios:

1. Guardar información médica personal.
2. Actualizar y eliminar registros médicos.
3. Acceder a su historial médico en cualquier momento.
4. Proteger la información mediante autenticación.

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.

```
// 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.room:room-runtime:2.3.0'
    annotationProcessor 'androidx.room:room-compiler:2.3.0'
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'
    implementation 'com.google.firebase:firebase-auth:20.0.4'
}
```

Paso 2: Diseño de la Interfaz de Usuario

Crear un archivo XML para la actividad principal (`activity_main.xml`) que incluya campos de entrada para la información médica, botones para guardar y eliminar registros, y una lista para mostrar el historial médico.

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

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Nombre"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/editTextAge"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Edad"
        android:inputType="number"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextName" />

    <EditText
        android:id="@+id/editTextCondition"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Condición Médica"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/editTextAge" />

    <Button
        android:id="@+id/buttonSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Guardar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextCondition"
/>

    <Button
        android:id="@+id/buttonDelete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Eliminar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonSave" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toBottomOf="@+id/buttonDelete"
        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 entrada de datos, guardar y eliminar registros utilizando la base de datos Room.

```
package com.example.medicalrecord;

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.EditText;
import android.widget.Toast;

import java.util.List;

public class MainActivity extends AppCompatActivity {

    private EditText editTextName;
```

```
    private EditText editTextAge;
    private EditText editTextCondition;
    private Button buttonSave;
    private Button buttonDelete;
    private RecyclerView recyclerView;
    private MedicalRecordViewModel medicalRecordViewModel;

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

        editTextName = findViewById(R.id.editTextName);
        editTextAge = findViewById(R.id.editTextAge);
        editTextCondition = findViewById(R.id.editTextCondition);
        buttonSave = findViewById(R.id.buttonSave);
        buttonDelete = findViewById(R.id.buttonDelete);
        recyclerView = findViewById(R.id.recyclerView);

        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setHasFixedSize(true);
        final MedicalRecordAdapter adapter = new
MedicalRecordAdapter();
        recyclerView.setAdapter(adapter);

        medicalRecordViewModel = new
ViewModelProvider(this).get(MedicalRecordViewModel.class);
        medicalRecordViewModel.getAllRecords().observe(this, new
Observer<List<MedicalRecord>>() {
            @Override
            public void onChanged(List<MedicalRecord> medicalRecords)
{
                adapter.setRecords(medicalRecords);
            }
        });

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

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

    private void saveRecord() {
        String name = editTextName.getText().toString();
        String age = editTextAge.getText().toString();
        String condition = editTextCondition.getText().toString();
```

```
        if (name.isEmpty() || age.isEmpty() || condition.isEmpty()) {
            Toast.makeText(this, "Por favor, complete todos los
campos", Toast.LENGTH_SHORT).show();
        } else {
            MedicalRecord record = new MedicalRecord(name,
Integer.parseInt(age), condition);
            medicalRecordViewModel.insert(record);
            Toast.makeText(this, "Registro guardado",
Toast.LENGTH_SHORT).show();
        }
    }

    private void deleteAllRecords() {
        medicalRecordViewModel.deleteAll();
        Toast.makeText(this, "Todos los registros eliminados",
Toast.LENGTH_SHORT).show();
    }
}
```

Paso 4: Configuración de Room

Crear las clases necesarias para la base de datos Room: `MedicalRecord.java`, `MedicalRecordDao.java`, `MedicalRecordDatabase.java` y `MedicalRecordRepository.java`.

```
// MedicalRecord.java
package com.example.medicalrecord;

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

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

    private String name;
    private int age;
    private String condition;

    public MedicalRecord(String name, int age, String condition) {
        this.name = name;
        this.age = age;
        this.condition = condition;
    }

    public void setId(int id) { this.id = id; }
    public int getId() { return id; }
    public String getName() { return name; }
    public int getAge() { return age; }
    public String getCondition() { return condition; }
}

// MedicalRecordDao.java
```

```
package com.example.medicalrecord;

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

import java.util.List;

@Dao
public interface MedicalRecordDao {
    @Insert
    void insert(MedicalRecord medicalRecord);

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

    @Query("SELECT * FROM medical_record_table ORDER BY name ASC")
    LiveData<List<MedicalRecord>> getAllRecords();
}

// MedicalRecordDatabase.java
package com.example.medicalrecord;

import android.content.Context;

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

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

    public abstract MedicalRecordDao medicalRecordDao();

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

// MedicalRecordRepository.java
package com.example.medicalrecord;

import android.app.Application;
```

```
import androidx.lifecycle.LiveData;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class MedicalRecordRepository {
    private MedicalRecordDao medicalRecordDao;
    private LiveData<List<MedicalRecord>> allRecords;
    private final ExecutorService executorService;

    public MedicalRecordRepository(Application application) {
        MedicalRecordDatabase database =
MedicalRecordDatabase.getInstance(application);
        medicalRecordDao = database.medicalRecordDao();
        allRecords = medicalRecordDao.getAllRecords();
        executorService = Executors.newFixedThreadPool(2);
    }

    public void insert(MedicalRecord medicalRecord) {
        executorService.execute(() ->
medicalRecordDao.insert(medicalRecord));
    }

    public void deleteAll() {
        executorService.execute(medicalRecordDao::deleteAll);
    }

    public LiveData<List<MedicalRecord>> getAllRecords() {
        return allRecords;
    }
}
```

Paso 5: Implementación del ViewModel

Crear una clase `MedicalRecordViewModel.java` para manejar los datos del historial médico y proporcionar métodos para interactuar con el repositorio.

```
package com.example.medicalrecord;

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

public class MedicalRecordViewModel extends AndroidViewModel {
    private MedicalRecordRepository repository;
    private LiveData<List<MedicalRecord>> allRecords;

    public MedicalRecordViewModel(@NonNull Application application) {
        super(application);
        repository = new MedicalRecordRepository(application);
        allRecords = repository.getAllRecords();
    }
```

```
    public void insert(MedicalRecord medicalRecord) {
        repository.insert(medicalRecord);
    }

    public void deleteAll() {
        repository.deleteAll();
    }

    public LiveData<List<MedicalRecord>> getAllRecords() {
        return allRecords;
    }
}
```

Paso 6: Implementación del Adaptador para RecyclerView

Crear una clase `MedicalRecordAdapter.java` para mostrar los registros médicos en un RecyclerView.

```
package com.example.medicalrecord;

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 MedicalRecordAdapter extends
RecyclerView.Adapter<MedicalRecordAdapter.MedicalRecordHolder> {
    private List<MedicalRecord> records = new ArrayList<>();

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

    @Override
    public void onBindViewHolder(@NonNull MedicalRecordHolder holder,
int position) {
        MedicalRecord currentRecord = records.get(position);
        holder.textViewName.setText(currentRecord.getName());

holder.textViewAge.setText(String.valueOf(currentRecord.getAge()));

holder.textViewCondition.setText(currentRecord.getCondition());
    }
```

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

    public void setRecords(List<MedicalRecord> records) {
        this.records = records;
        notifyDataSetChanged();
    }

    class MedicalRecordHolder extends RecyclerView.ViewHolder {
        private TextView textViewName;
        private TextView textViewAge;
        private TextView textViewCondition;

        public MedicalRecordHolder(View itemView) {
            super(itemView);
            textViewName = itemView.findViewById(R.id.text_view_name);
            textViewAge = itemView.findViewById(R.id.text_view_age);
            textViewCondition =
itemView.findViewById(R.id.text_view_condition);
        }
    }
}
```

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 3: Salud y Bienestar. A través de esta actividad, los estudiantes desarrollan habilidades en la gestión segura de datos, el uso de bases de datos Room y la creación de interfaces de usuario interactivas.

UAX Universidad Alfonso X el Sabio

G R A C I A S

UAX.COM