



**EXAMEN /** Programación Mobile

# Aprendizaje Esperado

En esta actividad logrará poner en práctica los conocimientos adquiridos durante toda la asignatura para construir aplicaciones para Android usando Android Studio, Kotlin, SQLite, Jetpack Compose y/o Vistas XML, la cámara y otros sensores del teléfono.

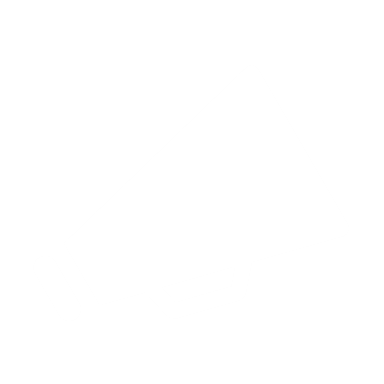
## INSTRUCCIONES GENERALES

Una vez realizada la lectura comprensiva de los materiales de estudio, es fundamental la realización de este examen, el que tiene por objetivo medir la correcta internalización y aplicación de los conceptos abordados.

El desarrollo de este trabajo es individual, no se permitirán entregas de talleres en duplas o en grupos

## CONTENIDOS ASOCIADOS AL EXAMEN:

* Interfaces de usuario usando Compose o Vistas XML
* Listas dinámicas (LazyColumn o RecyclerView)

****

**Esta actividad se complementa con una pauta de autoevaluación al finalizar, para que verifiques el logro**

**de tus aprendizajes**

* Persistencia en SQLite (ROOM o SQLiteOpenHelper)
* Consumo de Servicios Web
* Geolocalización y uso de la Cámara

**INSTRUCCIONES ESPECÍFICAS**

1. Cree las interfaces de usuario usando Compose o Vistas XML
2. A través de recursos permita que la aplicación sea multilenguaje. Los textos estáticos y textos de los botones deben estar traducidos al español y al inglés
3. Permita que su aplicación persista en SQLite los datos correspondientes del caso, utilice ROOM o SQLiteOpenHelper
4. Utilice Corrutinas para la programación asincrónica evitando bloquear la UI
5. Asegúrese de solicitar los permisos correspondientes ya sea a nivel de instalación o en tiempo de ejecución
6. Utilice los Web Services indicados para recuperar información desde internet
7. Aproveche la cámara y geolocalización para enriquecer las características de la aplicación solicitada
8. Documente el proceso realizado y suba a la plataforma lo desarrollado como se indica a continuación

**Se solicita:**

|  |
| --- |
| 1. Resolver el caso detallado a continuación 2. Documentar el proceso realizado explicando los pasos realizados y la lógica aplicada en un documento de Word (Fragmentos de código + Explicaciones) 3. Colocar enlace del código fuente del proyecto realizado ojalá en Github (o si no puede ser en Dropbox, Onedrive, Google Drive o similar) 4. Subir docx o doc a la plataforma (NO archivo comprimido) |

**Caso: App Vacaciones**

**Descripción del ejercicio:**

En este caso, los alumnos deberán desarrollar una aplicación para dispositivos Android que permita a los usuarios planificar sus vacaciones de manera eficiente. La aplicación deberá incluir características como la planificación de lugares a visitar en orden, la visualización de ubicaciones en un mapa utilizando osmdroid, la capacidad de agregar imágenes de referencia, tomar fotografías desde la aplicación, y la persistencia de datos en SQLite. Además, se debe integrar un servicio web para mostrar los costos en dólares en lugar de moneda local (originalmente en CLP).

**Requisitos Específicos:**

1. **Planificación de Lugares**: Los usuarios deben poder agregar lugares que desean visitar en su viaje. Cada lugar debe incluir información como el nombre del lugar, el orden de visita, una URL con una imagen de referencia del lugar (que se mostrará utilizando Coil), la latitud y longitud del lugar, el costo de alojamiento y el costo de transporte (si es aplicable), así como comentarios adicionales.
2. **Visualización de Mapa**: La aplicación debe utilizar el mapa de osmdroid para mostrar los lugares planificados y colocar marcadores en el mapa para cada ubicación.
3. **Agregar Fotos desde la Aplicación**: Los usuarios deben poder tomar al menos una fotografía desde la aplicación para cada lugar que visiten. Las fotos deben estar vinculadas al lugar correspondiente.
4. **Persistencia de Datos**: Todos los datos relacionados con los lugares, las imágenes de referencia y las fotografías tomadas deben ser almacenados en una base de datos SQLite en el dispositivo del usuario (para las imágenes almacenar sus URIs).
5. **Integración de Web Service**: La aplicación debe consumir un servicio web para obtener la tasa de cambio actual entre la moneda local (originalmente en CLP) y el dólar estadounidense. Esto se utilizará para mostrar los costos en dólares. Si no conoce ninguna, puede utilizar <https://mindicador.cl/api>
6. **Multilenguaje:** La aplicacióndebesoportar y estar traducida al inglés y al español (botones y textos estáticos).
7. **Interfaz de Usuario**: Construya a lo menos 3 pantallas: una para el listado de lugares, una para ver en detalle los datos del lugar y otra para editar los datos del lugar.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Figura 1 JSON con dólar observado desde API https//mindicador.cl/api

A continuación, imágenes referenciales (no es necesario que sean iguales) de la solución esperada:

|  |  |
| --- | --- |
| Figura 2 Vista con lista ordenada de lugares | Figura 3 Vista detalle lugar |

Texto

Descripción generada automáticamente con confianza media

Figura 4 Ejemplo de formulario

**Información Importante**

Esta actividad será evaluada por el profesor de la asignatura, por lo que se sugiere, para su realización, considerar los Criterios de Evaluación detallados al final de este documento.

El puntaje total de este taller es de 100 puntos, y el puntaje mínimo de aprobación es de 60 puntos.

**FORMALIDADES**

Tipo de letra: Arial 12

Interlineado: 1,15

* Formato - Entrega del taller en archivo Word, que lleve como nombre: SuNombre\_SuApellido.
* Subir archivo a la plataforma

**Pauta de Evaluación**

|  |  |  |
| --- | --- | --- |
| **Criterios de Evaluación** | **Indicadores de Logro** | **Puntaje** |
| Diseño de Interfaz de Usuario | Crea al menos 3 recursos de texto en inglés y español. Posteriormente los consume | 5 |
| Utiliza correctamente listas dinámicas (RecyclerView o LazyColumn) | 5 |
| Utiliza recursos gráficos como drawables o íconos para hacer la UI más amigable | 5 |
| Crea las pantallas necesarias para solucionar el caso | 5 |
| Persistencia de datos relevantes | Crea correctamente una data class para ROOM o clase para DTO si utiliza la API de SQLite directamente | 5 |
| Utiliza Corrutinas para evitar bloquear la UI cuando corresponde | 5 |
| Crea correctamente un método para conseguir los datos ordenados desde la BD | 5 |
| Crea correctamente un método para eliminar un registro de la BD | 5 |
| Consumo de Recursos y Servicios Web | Carga imagen externa a través de URL usando librería Coil | 5 |
| Configura correctamente la clase para mapear datos y la interfaz del servicio (Retrofit) | 10 |
| Consume correctamente los datos desde el web service | 5 |
| Geolocalización y Cámara | Muestra un mapa en base a latitud y longitud entregada por el usuario usando osmdroid | 5 |
| Solicita los permisos correspondientes para hacer funcionar la geolocalización y los mapas | 10 |
| Captura fotografías usando CameraX o solicitud a app externa | 5 |
| Documentación y Buenas Prácticas | Entrega en los plazos y forma (archivo docx) con enlace a código fuente | 5 |
| Documenta el proceso realizado para solucionar el problema. Explica los fragmentos de código más relevantes de manera clara | 10 |
| El código está ordenado, utiliza comentarios explicativos, utiliza nombres de variables y funciones significativos. | 5 |
| Puntaje Total | | 100 |

**Pauta de Autoevaluación:**

A continuación, te invitamos a responder una pauta de autoevaluación, para que verifiques lo aprendido en esta actividad y el logro del o los objetivos planteados inicialmente. Recuerda, en caso de presentar dudas, debes participar de forma permanente en los foros, donde el docente te retroalimentará y dará respuesta a tus inquietudes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterios de revisión del informe con la información recopilada:** | **Completamente logrado**  **(4)** | **Medianamente logrado**  **(3)** | **Parcialmente logrado**  **(2)** | **No logrado**  **(0)** |
| 1. Utilicé correctamente los recursos de texto para hacer mi app multilenguaje |  |  |  |  |
| 1. Utilicé recursos gráficos para hacer la interfaz más amigable y entendible |  |  |  |  |
| 1. Codifiqué todos los métodos necesarios para recuperar, crear, editar y eliminar registros desde la BD |  |  |  |  |
| 1. Utilicé correctamente Corrutinas para no bloquear la interfaz del dispositivo cuando era necesario |  |  |  |  |
| 1. Utilicé correctamente las listas dinámicas para mostrar los listados de información |  |  |  |  |
| 1. Capturé fotografías correctamente hacia un archivo |  |  |  |  |
| 1. Mostré mapas en base a coordenadas y coloqué un marcador en el punto correcto |  |  |  |  |
| 1. Consumí un servicio web usando Retrofit para realizar los cálculos solicitados |  |  |  |  |
| **Puntaje Total** | **32 puntos** | | | |
|  |  | | | |
| **Puntaje Obtenido** | **XX Puntos obtenidos** | | | |

**Examen**

# 2023-5B-PROGRAMACIÓN MOBILE/500-CED-ESPFT

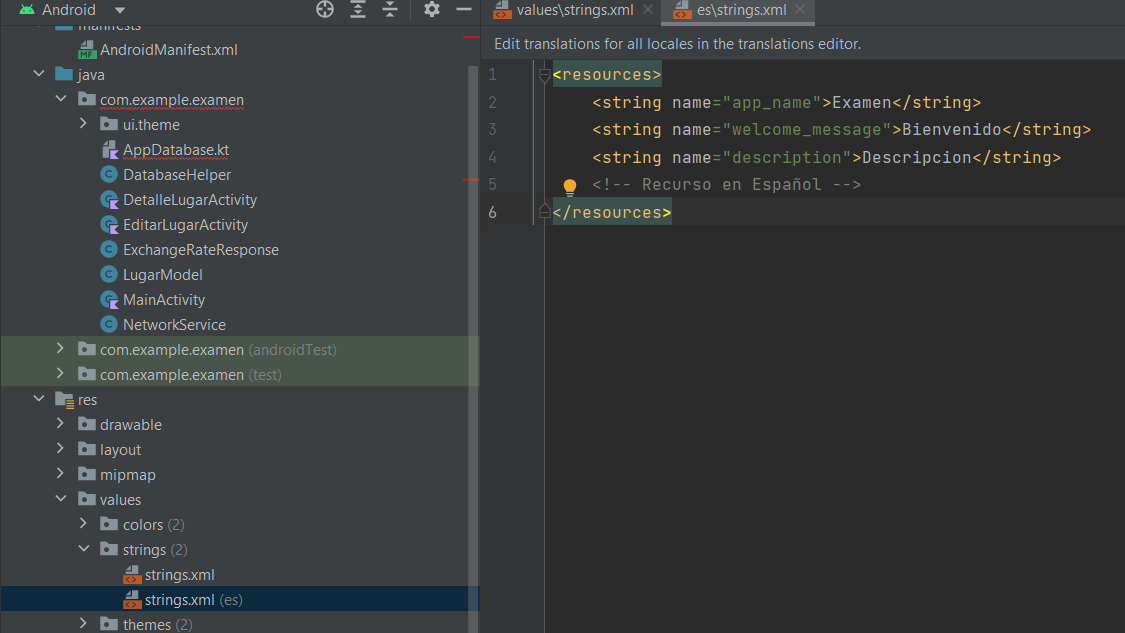
# Nataly Avila

Diseño de Interfaz de Usuario

1. Crea al menos 3 recursos de texto en inglés y español. Posteriormente los consume

<resources>  
 <string name="app\_name">Examen</string>  
 <string name="welcome\_message">Welcome</string>  
 <string name="description">description</string>  
 <!-- Recurso en Ingles -->  
</resources>

<resources>  
 <string name="app\_name">Examen</string>  
 <string name="welcome\_message">Bienvenido</string>  
 <string name="description">Descripcion</string>  
 <!-- Recurso en Español -->  
</resources>

****

2. Utiliza correctamente listas dinámicas (RecyclerView o LazyColumn)

private fun setupRecyclerView(lugares: List<Lugar>) {  
 val recyclerView: RecyclerView = findViewById(R.id.recycler\_view)  
 val dataList = obtenerDatos(lugares)  
 val adapter = MyAdapter(this, dataList)  
 recyclerView.adapter = adapter  
}

class MyAdapter  
 (  
 private val context: Context, private val dataList: List<String>  
) :  
 RecyclerView.Adapter<MyAdapter.ViewHolder?>() {

internal class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 var textView: TextView  
  
 init {  
 textView = itemView.findViewById<TextView>(R.id.text\_view)  
 }  
}

1. Utiliza recursos gráficos como drawables o íconos para hacer la UI más amigable

LazyColumn **{** items(viewModel.photos) **{** photoUri **->** PhotoThumbnail(imageName = "playa.png")  
 PhotoThumbnail(imageName = "termas.png")  
 **}  
}**

@Composable  
fun PhotoThumbnail(photoUri: Uri) {  
 Image(  
 painter = rememberImagePainter(photoUri),  
 contentDescription = "playa.png",  
 modifier = Modifier  
 .size(80.dp)  
 .padding(8.dp)  
 .clickable **{  
  
 }** )

1. Crea las pantallas necesarias para solucionar el caso

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <TextView  
 android:id="@+id/detalle\_text\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:padding="16dp"/>  
</RelativeLayout>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <EditText  
 android:id="@+id/editTextNombre"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:hint="Nombre del lugar"/>  
 <EditText  
 android:id="@+id/editTextDireccion"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/editTextNombre"  
 android:layout\_marginTop="8dp"  
 android:hint="Dirección del lugar"/>  
 <Button  
 android:id="@+id/buttonGuardar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/editTextDireccion"  
 android:layout\_marginTop="16dp"  
 android:text="Guardar cambios"/>  
</RelativeLayout>

Persistencia de datos relevantes

1. Crea correctamente una data class para ROOM o clase para DTO si utiliza la API de SQLite directamente

package com.example.examen  
  
import androidx.lifecycle.LiveData  
import androidx.lifecycle.MutableLiveData  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.*viewModelScope*import androidx.room.\*  
  
@Entity(tableName = "lugares")  
data class Lugar(  
 @PrimaryKey(autoGenerate = true)  
 val id: Long = 0,  
 val name: String,  
 val visitOrder: Int,  
 val imageUrl: String,  
 val latitude: Double,  
 val longitude: Double,  
 val accommodationCost: Double,  
 val transportCost: Double,  
 val additionalComments: String  
)  
  
@Dao  
interface LugarDao {  
 @Insert  
 suspend fun insertLugar(lugar: Lugar)  
  
 @Query("SELECT \* FROM lugares ORDER BY visitOrder")  
 suspend fun getOrderedLugares(): List<Lugar>  
  
 @Delete  
 suspend fun deleteLugar(lugar: Lugar)  
}  
  
@Database(entities = [Lugar::class], version = 1, exportSchema = false)  
abstract class AppDatabase : RoomDatabase() {  
 abstract fun lugarDao(): LugarDao  
}  
  
class AppRepository(private val lugarDao: LugarDao) {  
 suspend fun insertLugar(lugar: Lugar) {  
 lugarDao.insertLugar(lugar)  
 }  
  
 suspend fun getOrderedLugares(): List<Lugar> {  
 return lugarDao.getOrderedLugares()  
 }  
  
 suspend fun deleteLugar(lugar: Lugar) {  
 lugarDao.deleteLugar(lugar)  
 }  
}  
  
class LugarViewModel(private val repository: AppRepository) : ViewModel() {  
 private val \_lugares = MutableLiveData<List<Lugar>>()  
 val lugares: LiveData<List<Lugar>> get() = \_lugares  
  
 fun getOrderedLugares() {  
 *viewModelScope*.launch **{** \_lugares.*value* = repository.getOrderedLugares()  
 **}** }  
  
 fun insertLugar(lugar: Lugar) {  
 *viewModelScope*.launch **{** repository.insertLugar(lugar)  
 getOrderedLugares()  
 **}** }  
  
 fun deleteLugar(lugar: Lugar) {  
 *viewModelScope*.launch **{** repository.deleteLugar(lugar)  
 getOrderedLugares()  
 **}** }  
}

2. Utiliza Corrutinas para evitar bloquear la UI cuando corresponde

class AppRepository(private val lugarDao: LugarDao) {  
 suspend fun insertLugar(lugar: Lugar) {  
 lugarDao.insertLugar(lugar)  
 }  
  
 suspend fun getOrderedLugares(): List<Lugar> {  
 return lugarDao.getOrderedLugares()  
 }  
  
 suspend fun deleteLugar(lugar: Lugar) {  
 lugarDao.deleteLugar(lugar)  
 }  
}  
  
class LugarViewModel(private val repository: AppRepository) : ViewModel() {  
 private val \_lugares = MutableLiveData<List<Lugar>>()  
 val lugares: LiveData<List<Lugar>> get() = \_lugares  
  
 fun getOrderedLugares() {  
 *viewModelScope*.launch **{** \_lugares.*value* = repository.getOrderedLugares()  
 **}** }  
  
 fun insertLugar(lugar: Lugar) {  
 *viewModelScope*.launch **{** repository.insertLugar(lugar)  
 getOrderedLugares()  
 **}** }  
  
 fun deleteLugar(lugar: Lugar) {  
 *viewModelScope*.launch **{** repository.deleteLugar(lugar)  
 getOrderedLugares()  
 **}** }  
}

3. Crea correctamente un método para conseguir los datos ordenados desde la BD

@Dao  
interface LugarDao {  
 @Insert  
 suspend fun insertLugar(lugar: Lugar)  
  
 @Query("SELECT \* FROM lugares ORDER BY visitOrder")  
 suspend fun getOrderedLugares(): List<Lugar>  
  
 @Delete  
 suspend fun deleteLugar(lugar: Lugar)  
}

4.Crea correctamente un método para eliminar un registro de la BD

suspend fun deleteLugar(lugar: Lugar) {  
 lugarDao.deleteLugar(lugar)  
 }  
}

Consumo de Recursos y Servicios Web

1. Carga imagen externa a través de URL usando librería Coil

**build.gradle**

*implementation*("io.coil-kt:coil-compose:1.4.0")

@Composable  
fun PhotoThumbnail(photoUri: Uri) {  
 Image(  
 painter = rememberImagePainter(photoUri),  
 contentDescription = "photoUri",  
 modifier = Modifier  
 .size(80.dp)  
 .padding(8.dp)  
 .clickable **{  
  
 }** )  
}

2. Configura correctamente la clase para mapear datos y la interfaz del servicio (Retrofit)

package com.example.examen;  
  
import retrofit2.Retrofit;  
import retrofit2.converter.gson.GsonConverterFactory;  
  
public class NetworkService {  
  
 private static final String *BASE\_URL* = "https://mindicador.cl/api/";  
 private static NetworkService *instance*;  
 private Retrofit retrofit;  
  
 private NetworkService() {  
 retrofit = new Retrofit.Builder()  
 .baseUrl(*BASE\_URL*)  
 .addConverterFactory(GsonConverterFactory.create())  
 .build();  
 }  
  
 public static synchronized NetworkService getInstance() {  
 if (*instance* == null) {  
 *instance* = new NetworkService();  
 }  
 return *instance*;  
 }  
  
 public ExchangeRateService getExchangeRateService() {  
 return retrofit.create(ExchangeRateService.class);  
 }  
}

package com.example.examen;  
  
public class ExchangeRateResponse {  
 private String version;  
 private String Lugar;  
 private String dolar;  
  
 public String getVersion() {  
 return version;  
 }  
  
 public String getLugar() {  
 return lugar;  
 }  
  
 public String getDolar() {  
 return dolar;  
 }  
}

3. Consume correctamente los datos desde el web service

protected fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_detalle\_lugar.xml)  
 exchangeRateService = NetworkService.getInstance().exchangeRateService  
  
 val call: Call<ExchangeRateResponse> = exchangeRateService.getExchangeRate()  
 call.enqueue(object : Callback<ExchangeRateResponse?>() {  
 fun onResponse(  
 call: Call<ExchangeRateResponse?>?,  
 response: Response<ExchangeRateResponse?>  
 ) {  
 if (response.isSuccessful()) {  
 val exchangeRate: ExchangeRateResponse = response.body()  
 if (exchangeRate != null) {  
 Log.d("ExchangeRate", "Dólar: " + exchangeRate.*dolar*)  
 }  
 } else {  
 Log.e("NetworkError", "Error en la respuesta: " + response.code())  
 }  
 }  
  
 fun onFailure(call: Call<ExchangeRateResponse?>?, t: Throwable) {  
 Log.e("NetworkError", "Error en la solicitud: " + t.message)  
 }  
 })

Geolocalización y Cámara

1. Muestra un mapa en base a latitud y longitud entregada por el usuario usando osmdroid

fun OsmdroidMap() {  
 val context = *LocalContext*.current  
  
 Configuration.getInstance().load(context, context.getSharedPreferences("osmdroid", 0))  
  
 val mapView = remember **{** MapView(context) **}** mapView.setTileSource(TileSourceFactory.MAPNIK)  
  
 // Obtener la ubicación actual del ViewModel (asumo que tienes una propiedad location en viewModel)  
 val location = viewModel.location  
  
 if (location != null) {  
 mapView.controller.setCenter(GeoPoint(location.latitude, location.longitude))  
 mapView.controller.setZoom(15.0)  
 }  
  
 AndroidView(factory = **{** mapView **}**)  
}

2. Solicita los permisos correspondientes para hacer funcionar la geolocalización y los mapas

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContent **{** VacationApp(  
 viewModel = viewModel,  
 onCapturePhoto = **{** requestCameraPermission() **}**,  
 onOpenMap = **{** openMap() **}** )  
 **}** requestLocationPermission()  
}  
  
private fun requestLocationPermission() {  
 when {  
 ContextCompat.checkSelfPermission(  
 this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION* ) == PackageManager.*PERMISSION\_GRANTED* -> {  
 getLocation()  
 }  
 shouldShowRequestPermissionRationale(Manifest.permission.*ACCESS\_FINE\_LOCATION*) -> {  
 showPermissionRationale()  
 }  
 else -> {  
 requestPermissionLauncher.launch(Manifest.permission.*ACCESS\_FINE\_LOCATION*)  
 }  
 }  
}  
  
private fun showPermissionRationale() {  
 Toast.makeText(this, "Se necesitan permisos de ubicación para acceder a la ubicación actual.", Toast.*LENGTH\_LONG*).show()  
 requestPermissionLauncher.launch(Manifest.permission.*ACCESS\_FINE\_LOCATION*)  
}  
  
private val requestPermissionLauncher =  
 registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** isGranted **->** if (isGranted) {  
 getLocation()  
 } else {  
 showPermissionDeniedDialog()  
 }  
 **}**private fun getLocation() {  
 val context = *LocalContext*.current  
 val locationManager = context.getSystemService(Context.*LOCATION\_SERVICE*) as LocationManager  
  
 try {  
 if (locationManager.isProviderEnabled(LocationManager.*GPS\_PROVIDER*)) {  
 val locationCallback = object : LocationCallback() {  
 override fun onLocationResult(locationResult: LocationResult?) {  
 locationResult?.lastLocation?.let **{** viewModel.location = it  
 **}** locationManager.removeUpdates(this)  
 }  
 }  
 locationManager.requestLocationUpdates(  
 LocationManager.*GPS\_PROVIDER*,  
 0,  
 0f,  
 locationCallback,  
 context.*mainLooper* )  
 } else {  
 showLocationProviderDisabledDialog()  
 }  
 } catch (e: SecurityException) {  
 Toast.makeText(this, "Error al obtener la ubicación. Asegúrate de tener los permisos necesarios.", Toast.*LENGTH\_SHORT*).show()  
 }  
}

3.Captura fotografías usando CameraX o Solicitud a app externa

result?.let **{** viewModel.photos = viewModel.photos + it **}** getLocation()  
**}**@OptIn(ExperimentalPermissionsApi::class)  
private val locationPermissionState = rememberPermissionState(Manifest.permission.*ACCESS\_FINE\_LOCATION*)  
  
private val cameraPermissionState = rememberPermissionState(Manifest.permission.*CAMERA*)  
  
private val requestCameraPermissionLauncher =  
 registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** isGranted **->** if (isGranted) {  
 startCamera()  
 } else {  
 showCameraPermissionDeniedDialog()  
 }  
 **}**

private val cameraLauncher = registerForActivityResult(ActivityResultContracts.TakePicturePreview()) **{** result **->** result?.let **{** viewModel.photos = viewModel.photos + it **}** getLocation()  
**}**

private fun openMap() {  
 val location = viewModel.location  
  
 if (location != null) {  
 val uri = Uri.parse("geo:${location.latitude},${location.longitude}?z=15")  
 val intent = Intent(Intent.*ACTION\_VIEW*, uri)  
 intent.setPackage("com.google.android.apps.maps")  
  
 if (intent.resolveActivity(packageManager) != null) {  
 startActivity(intent)  
 } else {  
 Toast.makeText(this, "No se encontró una aplicación de mapas instalada", Toast.*LENGTH\_SHORT*).show()  
 }  
 } else {  
 Toast.makeText(this, "Ubicación no disponible", Toast.*LENGTH\_SHORT*).show()  
 }  
}

Documentación y Buenas Prácticas

En esta parte agrego las partes mas importantes del código de acorde a la funcionalidad de la aplicación

Archivo AndroidManifest.xml donde esta la configuración para los permisos de Camara e Internet

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
 <uses-feature android:name="android.hardware.camera" />  
 <uses-feature android:name="android.hardware.camera.autofocus" />  
  
 <uses-permission android:name="android.permission.CAMERA" />  
 <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
 <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
 <uses-permission android:name="android.permission.INTERNET" />  
  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.Examen"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:exported="true"  
 android:label="@string/app\_name"  
 android:theme="@style/Theme.Examen">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

Archivo build.gradle.kts

plugins **{** id("com.android.application")  
 *kotlin*("android")  
**}***android* **{** compileSdkVersion(34)  
  
 defaultConfig **{** applicationId = "com.example.evaluacion3"  
 minSdkVersion(24)  
 targetSdkVersion(34)  
 versionCode = 1  
 versionName = "1.0"  
 testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
 vectorDrawables.useSupportLibrary = true  
 namespace = "http://schemas.android.com/apk/res/com.example.evaluacion3"  
 **}** buildTypes **{** *release* **{** isMinifyEnabled = false  
 proguardFiles(getDefaultProguardFile("proguard-android-optimize.txt"), "proguard-rules.pro")  
 **}  
 }** compileOptions **{** sourceCompatibility = JavaVersion.*VERSION\_1\_8* targetCompatibility = JavaVersion.*VERSION\_1\_8* **}** *kotlinOptions* **{** jvmTarget = "1.8"  
 freeCompilerArgs += "-Xopt-in=kotlin.RequiresOptIn"  
 **}** buildFeatures **{** compose = true  
 viewBinding = true  
 **}** composeOptions **{** kotlinCompilerExtensionVersion = "1.0.5"  
 **}** packagingOptions **{** resources **{** excludes += "/META-INF/{AL2.0,LGPL2.1}"  
 **}  
 }  
}***dependencies* **{** *implementation*("androidx.core:core-ktx:1.7.0")  
 *implementation*("androidx.lifecycle:lifecycle-runtime-ktx:2.4.0")  
 *implementation*("androidx.activity:activity-compose:1.3.1")  
 *implementation*("androidx.compose.ui:ui:1.0.5")  
 *implementation*("androidx.compose.ui:ui-graphics:1.0.5")  
 *implementation*("androidx.compose.ui:ui-tooling-preview:1.0.5")  
 *implementation*("androidx.compose.material3:material3:1.0.0-beta01")  
 *testImplementation*("junit:junit:4.13.2")  
 *androidTestImplementation*("androidx.test.ext:junit:1.1.5")  
 *androidTestImplementation*("androidx.test.espresso:espresso-core:3.5.1")  
 *implementation*("androidx.camera:camera-core:1.1.0")  
 *implementation*("androidx.camera:camera-camera2:1.1.0")  
 *implementation*("androidx.camera:camera-lifecycle:1.1.0")  
 *implementation*("com.google.android.gms:play-services-location:18.0.0")  
 *implementation*("androidx.camera.core:camera-core:1.1.0-beta01")  
 *implementation*("androidx.camera.camera2:camera-camera2:1.1.0-beta01")  
 *implementation*("androidx.camera.lifecycle:camera-lifecycle:1.1.0-beta01")  
 *implementation*("androidx.camera.view:camera-view:1.1.0-alpha07")  
 *implementation*("org.osmdroid:osmdroid-android:6.1.10")  
 *implementation*("org.osmdroid:osmdroid-wms:6.1.10")  
 *implementation*("com.google.accompanist:accompanist-permissions:0.33.2-alpha")  
 *implementation*("com.google.accompanist:accompanist-insets:0.20.0")  
 *implementation*("io.coil-kt:coil-compose:1.4.0")  
**}**

Clase Lugar para ocupar Corrutinas

package com.example.examen  
  
import androidx.lifecycle.LiveData  
import androidx.lifecycle.MutableLiveData  
import androidx.lifecycle.ViewModel  
import androidx.lifecycle.*viewModelScope*import androidx.room.\*  
  
@Entity(tableName = "lugares")  
data class Lugar(  
 @PrimaryKey(autoGenerate = true)  
 val id: Long = 0,  
 val name: String,  
 val visitOrder: Int,  
 val imageUrl: String,  
 val latitude: Double,  
 val longitude: Double,  
 val accommodationCost: Double,  
 val transportCost: Double,  
 val additionalComments: String  
)  
  
@Dao  
interface LugarDao {  
 @Insert  
 suspend fun insertLugar(lugar: Lugar)  
  
 @Query("SELECT \* FROM lugares ORDER BY visitOrder")  
 suspend fun getOrderedLugares(): List<Lugar>  
  
 @Delete  
 suspend fun deleteLugar(lugar: Lugar)  
}  
  
@Database(entities = [Lugar::class], version = 1, exportSchema = false)  
abstract class AppDatabase : RoomDatabase() {  
 abstract fun lugarDao(): LugarDao  
}  
  
class AppRepository(private val lugarDao: LugarDao) {  
 suspend fun insertLugar(lugar: Lugar) {  
 lugarDao.insertLugar(lugar)  
 }  
  
 suspend fun getOrderedLugares(): List<Lugar> {  
 return lugarDao.getOrderedLugares()  
 }  
  
 suspend fun deleteLugar(lugar: Lugar) {  
 lugarDao.deleteLugar(lugar)  
 }  
}  
  
class LugarViewModel(private val repository: AppRepository) : ViewModel() {  
 private val \_lugares = MutableLiveData<List<Lugar>>()  
 val lugares: LiveData<List<Lugar>> get() = \_lugares  
  
 fun getOrderedLugares() {  
 *viewModelScope*.launch **{** \_lugares.*value* = repository.getOrderedLugares()  
 **}** }  
  
 fun insertLugar(lugar: Lugar) {  
 *viewModelScope*.launch **{** repository.insertLugar(lugar)  
 getOrderedLugares()  
 **}** }  
  
 fun deleteLugar(lugar: Lugar) {  
 *viewModelScope*.launch **{** repository.deleteLugar(lugar)  
 getOrderedLugares()  
 **}** }  
}

Cree un DatabaseHelper para poder trabajar con SQLite

package com.example.examen;  
  
import android.content.Context;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
  
public class DatabaseHelper extends SQLiteOpenHelper {  
 private static final String *DATABASE\_NAME* = "vacacion.db";  
 private static final int *DATABASE\_VERSION* = 1;  
  
 private static final String *TABLE\_PLACES* = "places";  
 private static final String *TABLE\_PHOTOS* = "photos";  
  
 // Common column names  
 private static final String *KEY\_ID* = "id";  
  
 // Places table columns  
 private static final String *KEY\_NAME* = "name";  
 private static final String *KEY\_VISIT\_ORDER* = "visit\_order";  
 private static final String *KEY\_IMAGE\_URL* = "image\_url";  
 private static final String *KEY\_LATITUDE* = "latitude";  
 private static final String *KEY\_LONGITUDE* = "longitude";  
 private static final String *KEY\_ACCOMMODATION\_COST* = "accommodation\_cost";  
 private static final String *KEY\_TRANSPORT\_COST* = "transport\_cost";  
 private static final String *KEY\_ADDITIONAL\_COMMENTS* = "additional\_comments";  
  
 // Photos table columns  
 private static final String *KEY\_PLACE\_ID* = "place\_id";  
 private static final String *KEY\_PHOTO\_URI* = "photo\_uri";  
  
 private static final String *CREATE\_TABLE\_PLACES* = "CREATE TABLE " + *TABLE\_PLACES* + "("  
 + *KEY\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT,"  
 + *KEY\_NAME* + " TEXT,"  
 + *KEY\_VISIT\_ORDER* + " INTEGER,"  
 + *KEY\_IMAGE\_URL* + " TEXT,"  
 + *KEY\_LATITUDE* + " REAL,"  
 + *KEY\_LONGITUDE* + " REAL,"  
 + *KEY\_ACCOMMODATION\_COST* + " REAL,"  
 + *KEY\_TRANSPORT\_COST* + " REAL,"  
 + *KEY\_ADDITIONAL\_COMMENTS* + " TEXT"  
 + ")";  
  
 private static final String *CREATE\_TABLE\_PHOTOS* = "CREATE TABLE " + *TABLE\_PHOTOS* + "("  
 + *KEY\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT,"  
 + *KEY\_PLACE\_ID* + " INTEGER,"  
 + *KEY\_PHOTO\_URI* + " TEXT"  
 + ")";  
  
 public DatabaseHelper(Context context) {  
 super(context, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 db.execSQL(*CREATE\_TABLE\_PLACES*);  
 db.execSQL(*CREATE\_TABLE\_PHOTOS*);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_PLACES*);  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_PHOTOS*);  
  
 onCreate(db);  
 }  
}

Archivo MainActivity.kt

package com.example.examen  
  
import android.Manifest  
import android.R  
import android.content.Context  
import android.content.Intent  
import android.content.pm.PackageManager  
import android.location.LocationManager  
import android.net.Uri  
import android.os.Bundle  
import android.provider.Settings  
import android.telecom.Call  
import android.util.Log  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.TextView  
import android.widget.Toast  
import androidx.activity.compose.setContent  
import androidx.activity.result.contract.ActivityResultContracts  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.fillMaxWidth  
import androidx.compose.foundation.layout.padding  
import androidx.compose.foundation.lazy.LazyColumn  
import androidx.compose.material3.Button  
import androidx.compose.material3.Icon  
import androidx.compose.material3.IconButton  
import androidx.compose.material3.Scaffold  
import androidx.compose.material3.Text  
import androidx.compose.material3.TopAppBar  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.DisposableEffect  
import androidx.compose.runtime.remember  
import androidx.compose.runtime.rememberUpdatedState  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.platform.*LocalContext*import androidx.compose.ui.viewinterop.AndroidView  
import androidx.core.content.ContextCompat  
import androidx.recyclerview.widget.RecyclerView  
import java.util.concurrent.ExecutorService  
import java.util.concurrent.Executors  
  
  
class MainActivity : AppCompatActivity() {  
 private var exchangeRateService: ExchangeRateService? = null  
  
 protected fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_detalle\_lugar.xml)  
 exchangeRateService = NetworkService.getInstance().exchangeRateService  
  
 val call: Call<ExchangeRateResponse> = exchangeRateService.getExchangeRate()  
 call.enqueue(object : Callback<ExchangeRateResponse?>() {  
 fun onResponse(  
 call: Call<ExchangeRateResponse?>?,  
 response: Response<ExchangeRateResponse?>  
 ) {  
 if (response.isSuccessful()) {  
 val exchangeRate: ExchangeRateResponse = response.body()  
 if (exchangeRate != null) {  
 Log.d("ExchangeRate", "Dólar: " + exchangeRate.*dolar*)  
 }  
 } else {  
 Log.e("NetworkError", "Error en la respuesta: " + response.code())  
 }  
 }  
  
 fun onFailure(call: Call<ExchangeRateResponse?>?, t: Throwable) {  
 Log.e("NetworkError", "Error en la solicitud: " + t.message)  
 }  
 })  
 }  
  
 private fun setupRecyclerView(lugares: List<Lugar>) {  
 val recyclerView: RecyclerView = findViewById(R.id.recycler\_view)  
 val dataList = obtenerDatos(lugares)  
 val adapter = MyAdapter(this, dataList)  
 recyclerView.adapter = adapter  
 }  
  
  
 private fun obtenerDatos(lugares: List<Lugar>): List<String> {  
 val dataList = *mutableListOf*<String>()  
 for (lugar in lugares) {  
 dataList.add("${lugar.nombre}, ${lugar.descripcion}")  
 }  
 return dataList  
 }  
  
  
 class MyAdapter  
 (  
 private val context: Context, private val dataList: List<String>  
 ) :  
 RecyclerView.Adapter<MyAdapter.ViewHolder?>() {  
  
  
 fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
 val view: View =  
 LayoutInflater.from(context).inflate(R.layout.item\_layout, parent, false)  
 return ViewHolder(view)  
 }  
  
  
 fun onBindViewHolder(holder: ViewHolder, position: Int) {  
 val item = dataList[position]  
 holder.textView.*text* = item  
 }  
  
  
 fun getItemCount(): Int {  
 return dataList.field  
 }  
  
  
 internal class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
 var textView: TextView  
  
 init {  
 textView = itemView.findViewById<TextView>(R.id.text\_view)  
 }  
 }  
 }  
 private val cameraLauncher = registerForActivityResult(ActivityResultContracts.TakePicturePreview()) **{** result **->** result?.let **{** viewModel.photos = viewModel.photos + it **}** getLocation()  
 **}** @OptIn(ExperimentalPermissionsApi::class)  
 private val locationPermissionState = rememberPermissionState(Manifest.permission.*ACCESS\_FINE\_LOCATION*)  
  
 private val cameraPermissionState = rememberPermissionState(Manifest.permission.*CAMERA*)  
  
 private val requestCameraPermissionLauncher =  
 registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** isGranted **->** if (isGranted) {  
 startCamera()  
 } else {  
 showCameraPermissionDeniedDialog()  
 }  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContent **{** VacationApp(  
 viewModel = viewModel,  
 onCapturePhoto = **{** requestCameraPermission() **}**,  
 onOpenMap = **{** openMap() **}** )  
 **}** requestLocationPermission()  
 }  
  
 private fun requestLocationPermission() {  
 when {  
 ContextCompat.checkSelfPermission(  
 this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION* ) == PackageManager.*PERMISSION\_GRANTED* -> {  
 getLocation()  
 }  
 shouldShowRequestPermissionRationale(Manifest.permission.*ACCESS\_FINE\_LOCATION*) -> {  
 showPermissionRationale()  
 }  
 else -> {  
 requestPermissionLauncher.launch(Manifest.permission.*ACCESS\_FINE\_LOCATION*)  
 }  
 }  
 }  
  
 private fun showPermissionRationale() {  
 Toast.makeText(this, "Se necesitan permisos de ubicación para acceder a la ubicación actual.", Toast.*LENGTH\_LONG*).show()  
 requestPermissionLauncher.launch(Manifest.permission.*ACCESS\_FINE\_LOCATION*)  
 }  
  
 private val requestPermissionLauncher =  
 registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** isGranted **->** if (isGranted) {  
 getLocation()  
 } else {  
 showPermissionDeniedDialog()  
 }  
 **}** private fun getLocation() {  
 val context = *LocalContext*.current  
 val locationManager = context.getSystemService(Context.*LOCATION\_SERVICE*) as LocationManager  
  
 try {  
 if (locationManager.isProviderEnabled(LocationManager.*GPS\_PROVIDER*)) {  
 val locationCallback = object : LocationCallback() {  
 override fun onLocationResult(locationResult: LocationResult?) {  
 locationResult?.lastLocation?.let **{** viewModel.location = it  
 **}** locationManager.removeUpdates(this)  
 }  
 }  
 locationManager.requestLocationUpdates(  
 LocationManager.*GPS\_PROVIDER*,  
 0,  
 0f,  
 locationCallback,  
 context.*mainLooper* )  
 } else {  
 showLocationProviderDisabledDialog()  
 }  
 } catch (e: SecurityException) {  
 Toast.makeText(this, "Error al obtener la ubicación. Asegúrate de tener los permisos necesarios.", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 private fun showLocationProviderDisabledDialog() {  
 Toast.makeText(this, "El proveedor de ubicación está desactivado. Habilita la ubicación desde la configuración del dispositivo.", Toast.*LENGTH\_LONG*).show()  
 }  
  
 private fun openMap() {  
 val location = viewModel.location  
  
 if (location != null) {  
 val uri = Uri.parse("geo:${location.latitude},${location.longitude}?z=15")  
 val intent = Intent(Intent.*ACTION\_VIEW*, uri)  
 intent.setPackage("com.google.android.apps.maps")  
  
 if (intent.resolveActivity(packageManager) != null) {  
 startActivity(intent)  
 } else {  
 Toast.makeText(this, "No se encontró una aplicación de mapas instalada", Toast.*LENGTH\_SHORT*).show()  
 }  
 } else {  
 Toast.makeText(this, "Ubicación no disponible", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 private fun showPermissionDeniedDialog() {  
 Toast.makeText(this, "Los permisos de ubicación están desactivados. Habilítalos manualmente desde la configuración de la aplicación.", Toast.*LENGTH\_LONG*).show()  
 startActivity(Intent(Settings.*ACTION\_APPLICATION\_DETAILS\_SETTINGS*, Uri.parse("package:$packageName")))  
 }  
  
 private fun showCameraPermissionDeniedDialog() {  
 Toast.makeText(this, "Los permisos de la cámara están desactivados. Habilítalos manualmente desde la configuración de la aplicación.", Toast.*LENGTH\_LONG*).show()  
 }  
  
 @Composable  
 fun VacationApp(viewModel: VacacionViewModel, onCapturePhoto: () -> Unit, onOpenMap: () -> Unit) {  
 val previewSurface = rememberUpdatedState(previewSurface) // Importante para asegurar la recreación de la vista previa al cambiar  
  
 Scaffold(  
 topBar = **{** TopAppBar(  
 title = **{** Text(text = "Registro de Vacaciones") **}**,  
 actions = **{** IconButton(  
 onClick = **{** onCapturePhoto()  
 **}** ) **{** Icon(imageVector = Icons.Default.PhotoCamera, contentDescription = "Capturar Foto")  
 **}  
 }** )  
 **}**,  
 content = **{** Column(  
 modifier = Modifier  
 .*fillMaxSize*()  
 .padding(16.dp)  
 ) **{** OutlinedTextField(  
 value = viewModel.placeName,  
 onValueChange = **{** viewModel.placeName = **it }**,  
 label = **{** Text("Nombre del lugar visitado") **}**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .padding(8.dp)  
 )  
 LazyColumn **{** items(viewModel.photos) **{** photoUri **->** PhotoThumbnail(imageName = "playa.png")  
 PhotoThumbnail(imageName = "termas.png")  
 **}  
 }** viewModel.location?.let **{** location **->** Text("Ubicación actual: Latitud ${location.latitude}, Longitud ${location.longitude}")  
 **}** ?: *run* **{** Text("Ubicación no disponible")  
 **}** Button(  
 onClick = **{** onOpenMap()  
 **}**,  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(vertical = 16.dp)  
 ) **{** Text("Ver en el mapa")  
 **}** CameraPreview(previewSurface.value)  
 **}  
 }** )  
 }  
  
 @Composable  
 fun PhotoThumbnail(photoUri: Uri) {  
 Image(  
 painter = rememberImagePainter(photoUri),  
 contentDescription = "photoUri",  
 modifier = Modifier  
 .size(80.dp)  
 .padding(8.dp)  
 .clickable **{  
  
 }** )  
 }  
 @Composable  
 fun CameraPreview(previewSurface: Preview.SurfaceProvider) {  
 val context = *LocalContext*.current  
 val cameraProviderFuture = ProcessCameraProvider.getInstance(context)  
 var camera: Camera? = null  
 val cameraExecutor: ExecutorService = Executors.newSingleThreadExecutor()  
  
 AndroidView(  
 modifier = Modifier.*fillMaxWidth*().height(300.dp),  
 factory = **{** context **->** val view = PreviewView(context)  
 val cameraProvider = cameraProviderFuture.get()  
 val cameraSelector = CameraSelector.DEFAULT\_BACK\_CAMERA  
 val preview = Preview.Builder()  
 .setTargetAspectRatio(AspectRatio.RATIO\_4\_3)  
 .build()  
 preview.setSurfaceProvider(previewSurface)  
 val imageCapture = ImageCapture.Builder()  
 .build()  
  
 try {  
 camera = cameraProvider.bindToLifecycle(  
 context as LifecycleOwner,  
 cameraSelector,  
 preview,  
 imageCapture  
 )  
 } catch (exc: Exception) {  
  
 }  
  
 view  
 **}**,  
 )  
  
 DisposableEffect(context) **{** onDispose **{** cameraExecutor.shutdown()  
 **}  
 }** Composable  
 fun OsmdroidMap() {  
 val context = *LocalContext*.current  
  
 Configuration.getInstance().load(context, context.getSharedPreferences("osmdroid", 0))  
  
 val mapView = remember **{** MapView(context) **}** mapView.setTileSource(TileSourceFactory.MAPNIK)  
  
 Scaffold(  
 content = **{** AndroidView(factory = **{** mapView **}**)  
 **}** )  
 }  
  
 fun OsmdroidMap() {  
 val context = *LocalContext*.current  
  
 Configuration.getInstance().load(context, context.getSharedPreferences("osmdroid", 0))  
  
 val mapView = remember **{** MapView(context) **}** mapView.setTileSource(TileSourceFactory.MAPNIK)  
  
 // Obtener la ubicación actual del ViewModel (asumo que tienes una propiedad location en viewModel)  
 val location = viewModel.location  
  
 if (location != null) {  
 mapView.controller.setCenter(GeoPoint(location.latitude, location.longitude))  
 mapView.controller.setZoom(15.0)  
 }  
  
 AndroidView(factory = **{** mapView **}**)  
 }  
 }

