UNIVERSIDAD DON BOSCO

Docente:

Julio Armando García Fabián

Asignatura:

Desarrollo de Software para Moviles

Integrantes:

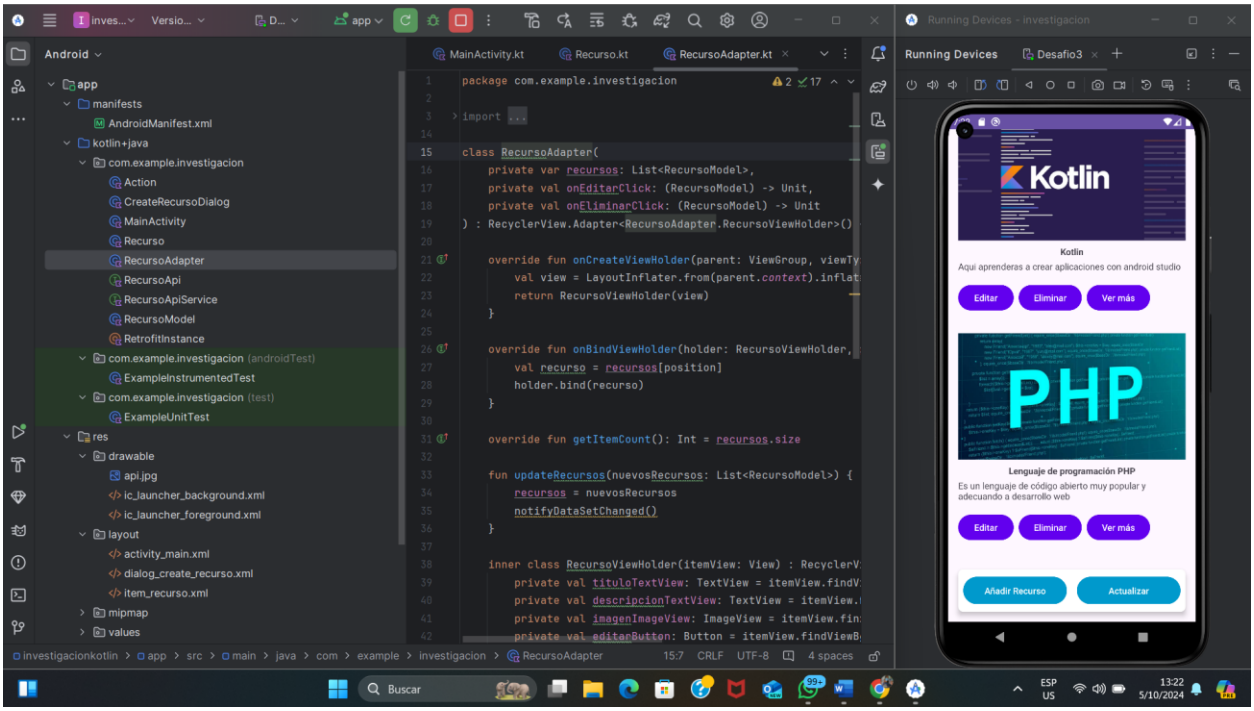
Cristian Ricardo González González (GG222955)

Dania Gorety Tejada Campos (TC232020)

Grupo:

G01L

Desafío 3

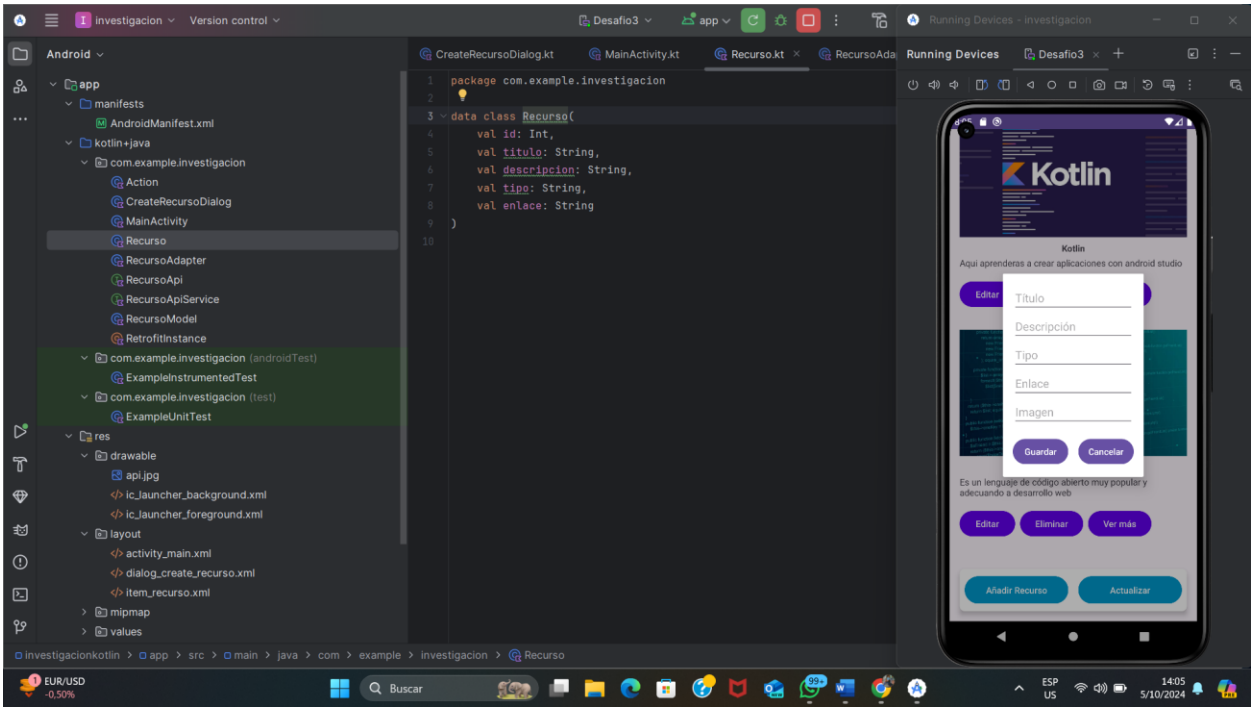


Código fuente:

```
package com.example.investigacion

import android.content.Context
import android.content.Intent
import android.net.Uri
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide

class RecursoAdapter(
    private var recursos: List<RecursoModel>,
    private val onEditarClick: (RecursoModel) -> Unit,
    private val onEliminarClick: (RecursoModel) -> Unit
) : RecyclerView.Adapter<RecursoAdapter.RecursoViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
```

```
RecursoViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_recurso, parent,
false)
        return RecursoViewHolder(view)
    }

    override fun onBindViewHolder(holder: RecursoViewHolder, position: Int) {
        val recurso = recursos[position]
        holder.bind(recurso)
    }

    override fun getItemCount(): Int = recursos.size

    fun updateRecursos(nuevosRecursos: List<RecursoModel>) {
        recursos = nuevosRecursos
        notifyDataSetChanged()
    }

    inner class RecursoViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private val tituloTextView: TextView = itemView.findViewById(R.id.tv_titulo)
        private val descripcionTextView: TextView =
itemView.findViewById(R.id.tv_descripcion)
        private val imagenImageView: ImageView =
itemView.findViewById(R.id.iv_imagen)
        private val editarButton: Button = itemView.findViewById(R.id.btn_editar)
        private val eliminarButton: Button = itemView.findViewById(R.id.btn_eliminar)
        private val verMasButton: Button = itemView.findViewById(R.id.btn_ver_mas)

        fun bind(recurso: RecursoModel) {
            tituloTextView.text = recurso.titulo
            descripcionTextView.text = recurso.descripcion

            // Cargar la imagen usando Glide
            Glide.with(itemView.context)
                .load(recurso.imagen)
                .into(imagenImageView)

            editarButton.setOnClickListener {
                onEditarClick(recurso)
            }

            eliminarButton.setOnClickListener {
```

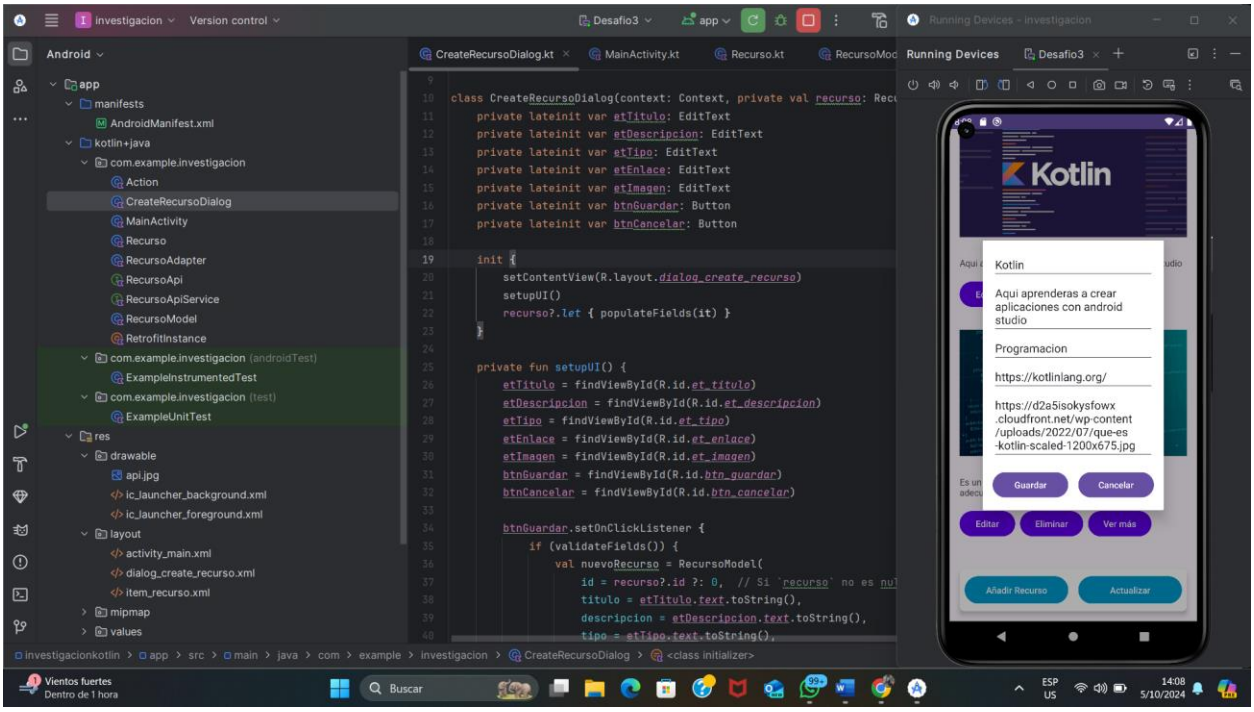
```
                onEliminarClick(recurso)
            }

            verMasButton.setOnClickListener {
                val intent = Intent(Intent.ACTION_VIEW, Uri.parse(recurso.enlace))
                itemView.context.startActivity(intent)
            }
        }
    }
}
```



Código fuente:

```
package com.example.investigacion

data class Recurso(
    val id: Int,
    val titulo: String,
    val descripcion: String,
    val tipo: String,
    val enlace: String
)
```

```
package com.example.investigacion

data class RecursoModel(
    val id: Long,
    val titulo: String,
    val descripcion: String,
    val tipo: String,
    val enlace: String,
    val imagen: String
)
```

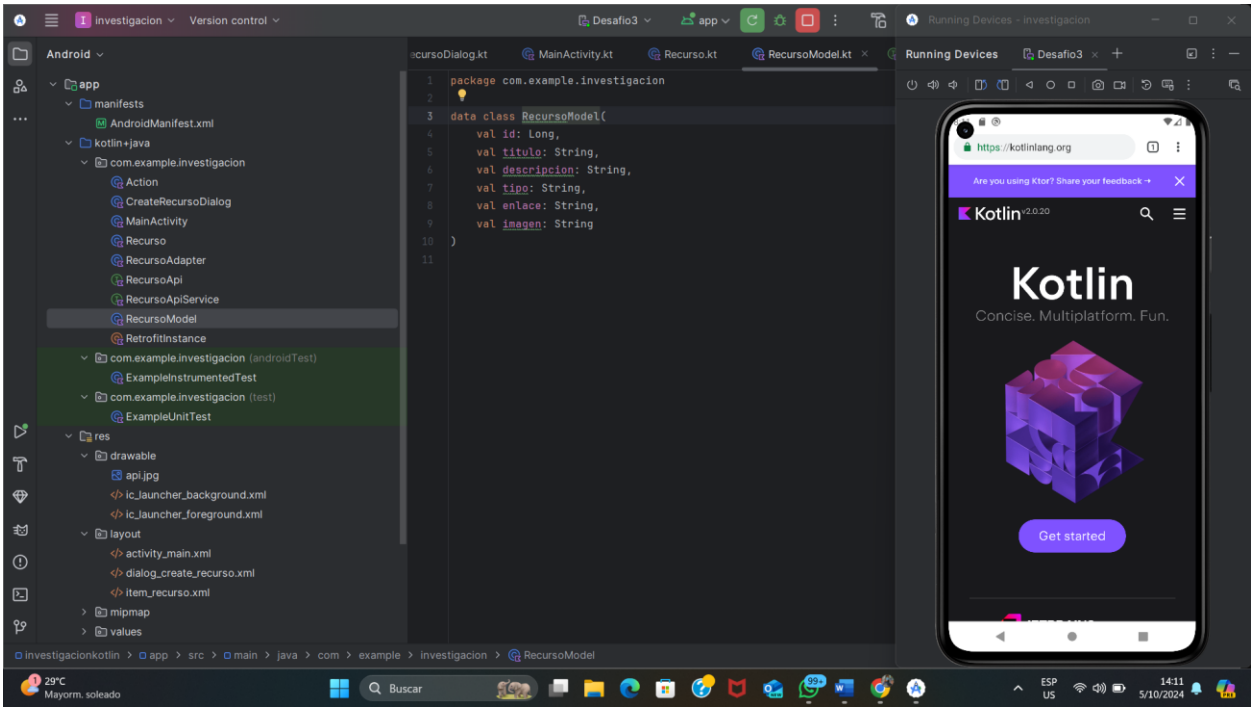


Código Fuente:

```
package com.example.investigacion

import android.app.Dialog
import android.content.Context
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.Toast

class CreateRecursoDialog(context: Context, private val recurso: RecursoModel? = null,
private val onSave: (RecursoModel) -> Unit) : Dialog(context) {
    private lateinit var etTitulo: EditText
    private lateinit var etDescripcion: EditText
```

```
    private lateinit var etTipo: EditText
    private lateinit var etEnlace: EditText
    private lateinit var etImagen: EditText
    private lateinit var btnGuardar: Button
    private lateinit var btnCancelar: Button

    init {
        setContentView(R.layout.dialog_create_recurso)
        setupUI()
        recurso?.let { populateFields(it) }
    }

    private fun setupUI() {
        etTitulo = findViewById(R.id.et_titulo)
        etDescripcion = findViewById(R.id.et_descripcion)
        etTipo = findViewById(R.id.et_tipo)
        etEnlace = findViewById(R.id.et_enlace)
        etImagen = findViewById(R.id.et_imagen)
        btnGuardar = findViewById(R.id.btn_guardar)
        btnCancelar = findViewById(R.id.btn_cancelar)

        btnGuardar.setOnClickListener {
            if (validateFields()) {
                val nuevoRecurso = RecursoModel(
                    id = recurso?.id ?: 0,  // Si `recurso` no es nulo, utiliza su ID; si es nulo, usa
0
                    titulo = etTitulo.text.toString(),
                    descripcion = etDescripcion.text.toString(),
                    tipo = etTipo.text.toString(),
                    enlace = etEnlace.text.toString(),
                    imagen = etImagen.text.toString()
                )
                onSave(nuevoRecurso)
                dismiss()
            }
        }

        btnCancelar.setOnClickListener {
            dismiss()
        }
    }

    private fun populateFields(recurso: RecursoModel) {
```

```
        etTitulo.setText(recurso.titulo)
        etDescripcion.setText(recurso.descripcion)
        etTipo.setText(recurso.tipo)
        etEnlace.setText(recurso.enlace)
        etImagen.setText(recurso.imagen)
    }

    private fun validateFields(): Boolean {
        if (etTitulo.text.isEmpty()) {
            Toast.makeText(context, "Por favor ingresa un título",
Toast.LENGTH_SHORT).show()
            return false
        }
        return true
    }
}
```




```
package com.example.investigacion

import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.ProgressBar
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
```

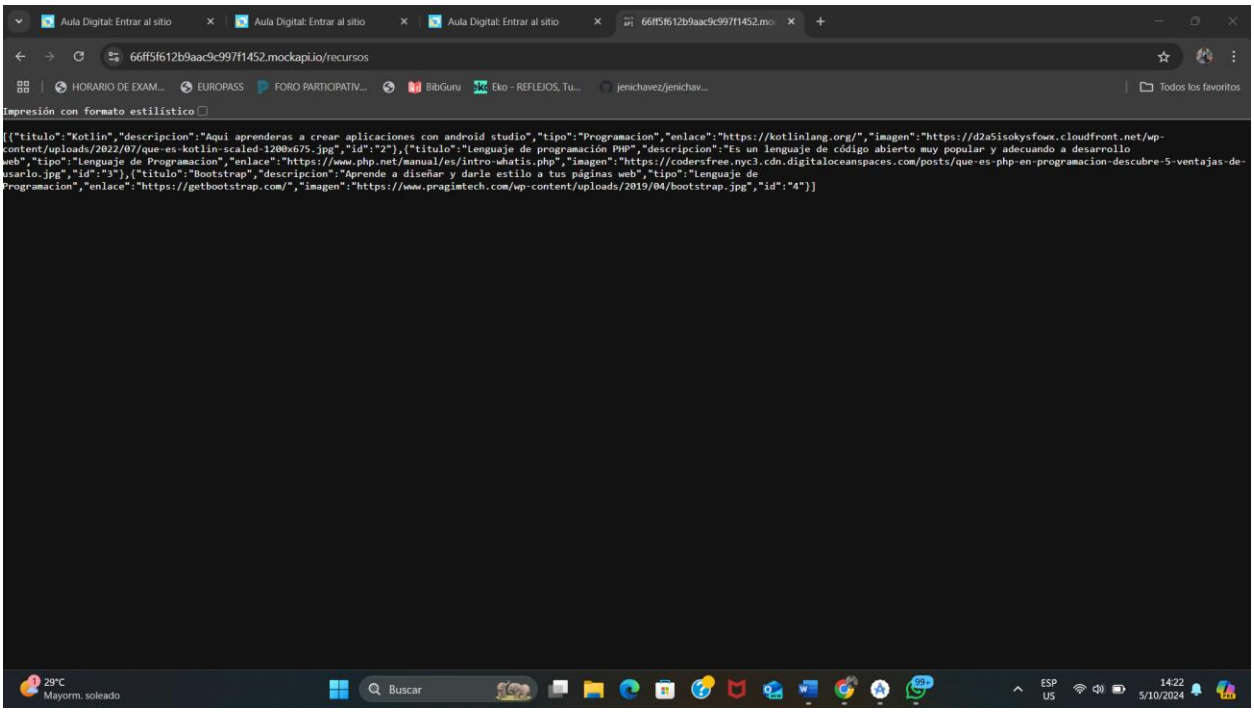
```
import androidx.lifecycle.lifecycleScope
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import kotlinx.coroutines.launch
import retrofit2.HttpException
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import java.io.IOException

class MainActivity : AppCompatActivity() {
    private val urlBase = "https://66ff5f612b9aac9c997f1452.mockapi.io/"
    private lateinit var recyclerView: RecyclerView
    private lateinit var adapter: RecursoAdapter
    private lateinit var progressBar: ProgressBar

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        setupUI()
        fetchAndDisplayRecursos()
    }

    private fun setupUI() {
        recyclerView = findViewById(R.id.rv_recursos)
        progressBar = findViewById(R.id.progress_bar)

        recyclerView.layoutManager = LinearLayoutManager(this)
        adapter = RecursoAdapter(emptyList(), ::onEditRecurso, ::onDeleteRecurso)
        recyclerView.adapter = adapter

        val refreshButton: Button = findViewById(R.id.btn_refresh)
        refreshButton.setOnClickListener {
            fetchAndDisplayRecursos()
        }

        val addButton: Button = findViewById(R.id.btn_add_recurso)
        addButton.setOnClickListener {
            showCreateRecursoDialog()
        }
    }

    private fun fetchAndDisplayRecursos() {
```

```
        progressBar.visibility = View.VISIBLE
        recyclerView.visibility = View.GONE

        val retrofit = Retrofit.Builder()
            .baseUrl(urlBase)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        val service = retrofit.create(RecursoApiService::class.java)

        lifecycleScope.launch {
            try {
                val recursos = service.getRecursos()
                adapter.updateRecursos(recursos)
                recyclerView.visibility = View.VISIBLE
                if (recursos.isEmpty()) {
                    Toast.makeText(this@MainActivity, "No se encontraron recursos",
Toast.LENGTH_SHORT).show()
                }
            } catch (e: Exception) {
                val message = when (e) {
                    is IOException -> "Error de red: verifica tu conexión a internet"
                    is HttpException -> "Error del servidor: ${e.code()}"
                    else -> "Error: ${e.message}"
                }
                Toast.makeText(this@MainActivity, message, Toast.LENGTH_LONG).show()
            } finally {
                progressBar.visibility = View.GONE
            }
        }
    }

    private fun showCreateRecursoDialog() {
        val dialog = CreateRecursoDialog(this) { nuevoRecurso ->
            crearNuevoRecurso(nuevoRecurso)
        }
        dialog.show()
    }

    private fun onEditRecurso(recurso: RecursoModel) {
        val dialog = CreateRecursoDialog(this, recurso) { updatedRecurso ->
            actualizarRecurso(updatedRecurso)
        }
        dialog.show()
```

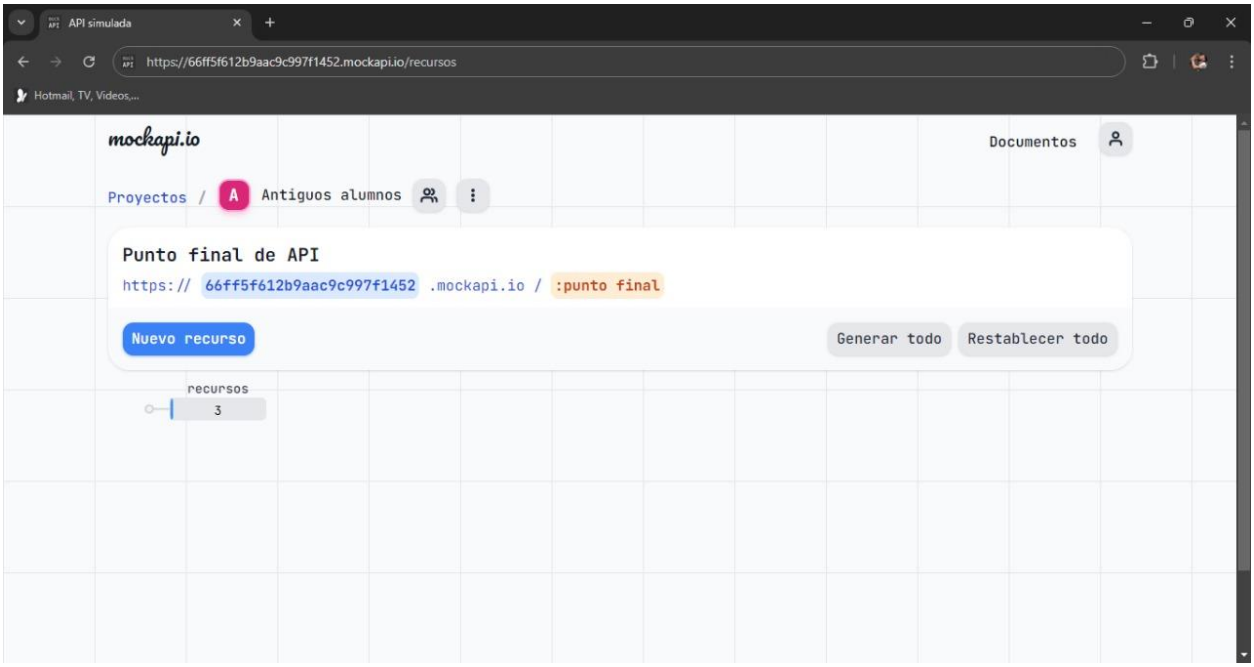
```
    }

    private fun onDeleteRecurso(recurso: RecursoModel) {
        eliminarRecurso(recurso)
    }

    private fun crearNuevoRecurso(nuevoRecurso: RecursoModel) {
        val retrofit = Retrofit.Builder()
            .baseUrl(urlBase)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        val service = retrofit.create(RecursoApiService::class.java)

        lifecycleScope.launch {
            try {
                val recursoCreado = service.createRecurso(nuevoRecurso)
                Toast.makeText(this@MainActivity, "Recurso creado",
Toast.LENGTH_SHORT).show()
                fetchAndDisplayRecursos() // Actualiza la lista de recursos
            } catch (e: Exception) {
                Toast.makeText(this@MainActivity, "Error: ${e.message}",
Toast.LENGTH_SHORT).show()
            }
        }
    }

    private fun actualizarRecurso(recurso: RecursoModel) {
        val retrofit = Retrofit.Builder()
            .baseUrl(urlBase)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        val service = retrofit.create(RecursoApiService::class.java)

        lifecycleScope.launch {
            try {
                service.updateRecurso(recurso.id, recurso)
                Toast.makeText(this@MainActivity, "Recurso actualizado",
Toast.LENGTH_SHORT).show()
                fetchAndDisplayRecursos() // Actualiza la lista de recursos
            } catch (e: Exception) {
                Toast.makeText(this@MainActivity, "Error: ${e.message}",
Toast.LENGTH_SHORT).show()
            }
```

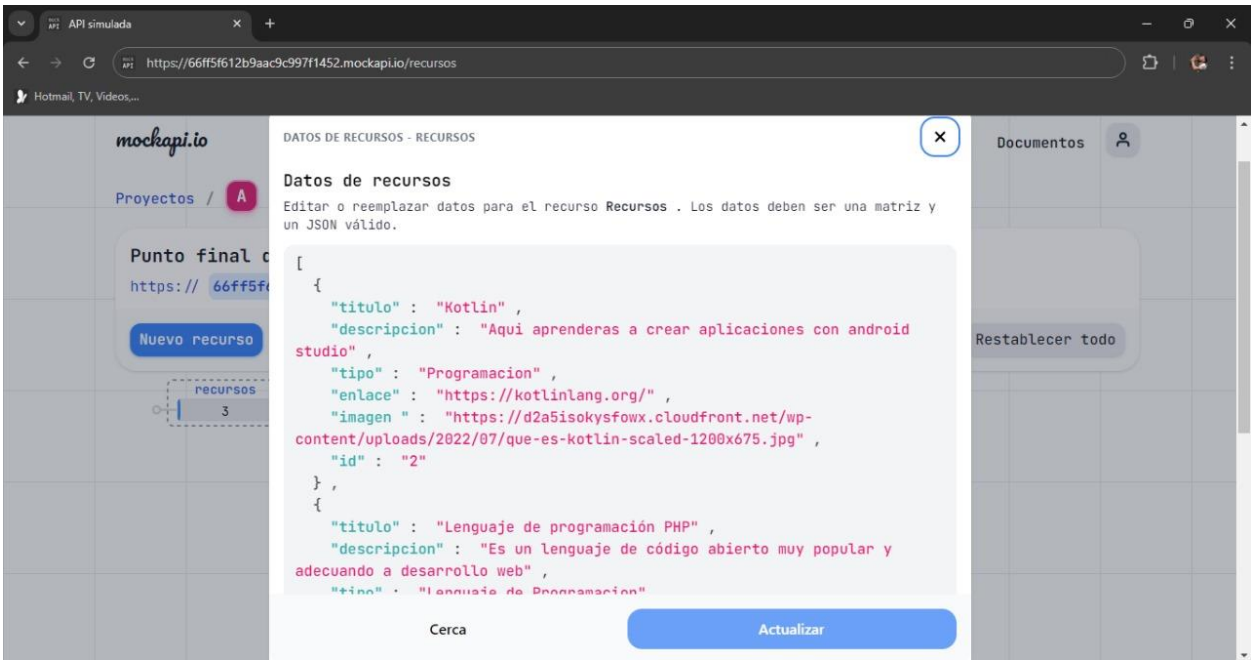
```
        }
    }

    private fun eliminarRecurso(recurso: RecursoModel) {
        val retrofit = Retrofit.Builder()
            .baseUrl(urlBase)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        val service = retrofit.create(RecursoApiService::class.java)

        lifecycleScope.launch {
            try {
                service.deleteRecurso(recurso.id)
                Toast.makeText(this@MainActivity, "Recurso eliminado",
Toast.LENGTH_SHORT).show()
                fetchAndDisplayRecursos() // Actualiza la lista de recursos
            } catch (e: Exception) {
                Toast.makeText(this@MainActivity, "Error: ${e.message}",
Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

Api



```
package com.example.investigacion

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

object RetrofitInstance {
    private const val BASE_URL = "https://66ff5f612b9aac9c997f1452.mockapi.io/"

    val api: RecursoApi by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(RecursoApi::class.java)
    }
}
```






```
package com.example.investigacion

import retrofit2.Call
import retrofit2.http.Body
import retrofit2.http.DELETE
import retrofit2.http.GET
import retrofit2.http.POST
import retrofit2.http.PUT
import retrofit2.http.Path

interface RecursoApiService {
    @GET("recursos")
    suspend fun getRecursos(): List<RecursoModel>

    @POST("recursos")
    suspend fun createRecurso(@Body recurso: RecursoModel): RecursoModel

    @PUT("recursos/{id}")
    suspend fun updateRecurso(@Path("id") id: Long, @Body recurso: RecursoModel)

    @DELETE("recursos/{id}")
    suspend fun deleteRecurso(@Path("id") id: Long)
}
```